DeepObfuscator: Obfuscating Intermediate Representations with Privacy-Preserving Adversarial Learning on Smartphones

Ang Li[1], Jiayi Guo[2], Huanrui Yang[1], Flora D. Salim[3], Yiran Chen[1]
[1]Department of Electrical and Computer Engineering, Duke University
[2]Department of Automation, Tsinghua University
[3]Computer Science and Information Technology, School of Science, RMIT University
[1]{ang.li630, huanrui.yang, yiran.chen}@duke.edu, [2]guo-jy20@mails.tsinghua.edu.cn, [3]flora.salim@rmit.edu.au

ABSTRACT

Deep learning has been widely applied in many computer vision applications, with remarkable success. However, running deep learning models on mobile devices is generally challenging due to the limitation of computing resources. A popular alternative is to use cloud services to run deep learning models to process raw data. This, however, imposes privacy risks. Some prior arts proposed sending the features extracted from raw data (e.g., images) to the cloud. Unfortunately, these extracted features can still be exploited by attackers to recover raw images and to infer embedded private attributes (e.g., age, gender, etc.). In this paper, we propose an adversarial training framework, *DeepObfuscator*, which prevents the usage of the features for reconstruction of the raw images and inference of private attributes. This is done while retaining useful information for the intended cloud service (i.e., image classification). DeepObfuscator includes a learnable encoder, namely, obfuscator that is designed to hide privacy-related sensitive information from the features by performing our proposed adversarial training algorithm. The proposed algorithm is designed by simulating the game between an attacker who makes efforts to reconstruct raw image and infer private attributes from the extracted features and a defender who aims to protect user privacy. By deploying the trained obfuscator on the smartphone, features can be locally extracted and then sent to the cloud. Our experiments on CelebA and LFW datasets show that the quality of the reconstructed images from the obfuscated features of the raw image is dramatically decreased from 0.9458 to 0.3175 in terms of multi-scale structural similarity (MS-SSIM). The person in the reconstructed image, hence, becomes hardly to be re-identified. The classification accuracy of the inferred private attributes that can be achieved by the attacker is significantly reduced to a random-guessing level, e.g., the accuracy of gender is reduced from 97.36% to 58.85%. As a comparison, the accuracy of the intended classification tasks performed via the cloud service is only reduced by 2%. We also demonstrate the efficiency of DeepObfuscator, showcasing real-time performance of the deployed models on smartphones.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; • **Security and privacy** → **Usability in security and privacy**.

KEYWORDS

Privacy, Adversarial Learning, Smartphones

1 INTRODUCTION

In the past decade, deep learning has achieved great success in many computer vision applications, such as face recognition [29] and image segmentation [18]. However, running deep learning models on mobile devices is technically challenging due to limited computing resources. Many large-sized deep-learning-based applications are often deployed on cloud servers, i.e., ML-as-a-service (MLaaS), such as Amazon Rekognition, Microsoft Cognitive Services, etc. These cloud-based services require users to send data (e.g., images) to the cloud service provider. However, this requirement may raise users' concerns about privacy leakage, since various private information may be contained in the images (e.g., age, gender, etc.). One widely adopted solution to address this privacy issue is to upload only the extracted features rather than the raw image [26, 27]. Unfortunately, the extracted features still contain rich information which can breach users' privacy. Specifically, an attacker can exploit the eavesdropped features to reconstruct the raw image, and hence the identity of the person on the raw image can be discovered from the reconstructed image [21]. In addition, the extracted features can also be exploited by an attacker to infer private attributes, such as gender, age, etc. Such adversary models can be trained by an attacker through continuously querying the cloud service to collect the eavesdropped features as inputs, and the ground truth of the queried data can be used as the labels.

Figure 1 shows an example where the reconstruction attack and private attribute leakage occur in a MLaaS for facial attribute recognition. When a mobile user uploads an image for detecting facial attribute, the encoder will extract features and then send the features to the cloud server. The classifier deployed on the server takes the extracted features as inputs and then predicts whether the person in the received image is smiling or not. Note that a complete model is jointly trained in an end-to-end manner, and then split into the encoder and the classifier. However, the extracted features can be eavesdropped by an attacker. By continuously querying the cloud service, the attacker can collect the eavesdropped features to train an decoder, which is denoted as the adversary reconstructor, for recovering the raw images. Besides, the eavesdropped features can also be exploited to train an adversary classifier for inferring the private attributes associated with the raw images.

There are a few studies have been performed to defend against reconstruction attacks. They perturbed either the raw data [10] or the extracted features [26, 27] through adding random noises. But these methods inevitably incur accuracy drop. Feutry *et al.* [6] proposed an image anonymization approach to hide sensitive features related to private attributes. However, none of these previous studies has investigated whether defending against only reconstruction

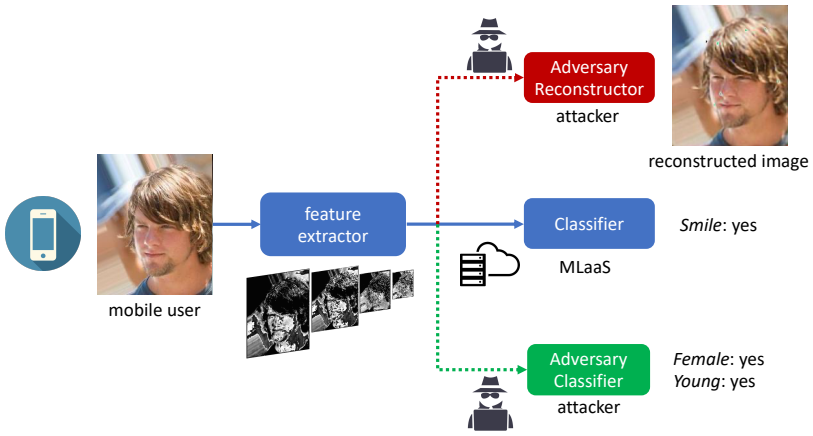


Figure 1: An example of reconstruction attack and private attribute leakage in a cloud service for facial attribute recognition.

attack or private attribute leakage is sufficient to prevent either or both types of attacks.

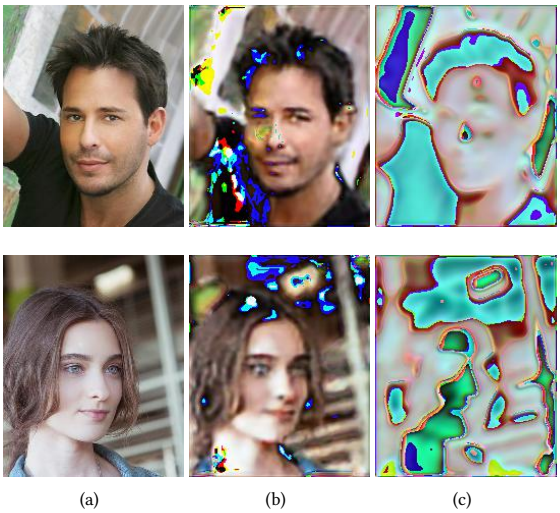


Figure 2: Reconstructed images: defending against only private attribute leakage vs. defending against only reconstruction attack. Column (a) is raw images, column (b) shows the reconstructed images when we defend against only private attribute leakage, and column (c) displays reconstructed images when defending against only reconstruction attack.

Our experiments demonstrate that defending only reconstruction attack cannot prevent private attribute leakage, and vice versa. For example, Figure 2 shows two examples of reconstructed images when we defend against only either the reconstruction attack or private attribute leakage. The column (a) shows the raw images, and the column (b) displays the reconstructed images when we defend only private attribute leakage. These reconstructed images still contain many details of the raw images, and allow an attacker to re-identify the person in the images. The reconstructed images when defending against only reconstruction attack are shown in the column (c). Although almost all distinguishable information has been masked, we can still achieve a 93.7% accuracy in detecting the gender of the person in the reconstructed images. More details about this experiment can be found in Section 4.2.

In this work, we propose DeepObfuscator – an adversarial training framework to learn an obfuscator that can hide sensitive information that can be exploited for reconstructing raw images and inferring private attributes, and still keep useful features for image classifications. Although we focus on image classifications in this paper, DeepObfuscator can be easily extended to many other tasks, e.g., speech recognition. By deploying the trained obfuscator on the smartphone, features can be locally extracted and then sent to MLaaS provider. As Figure 3 shows, DeepObfuscator consists of four modules: *obfuscator*, *classifier*, *adversary reconstructor* and *adversary classifier*. The key idea is to apply adversarial training for maximizing the reconstruction error of the adversary reconstructor and the classification error of the adversary classifier, but minimizing the classification error of the intended classifier.

The main contributions of this paper are summarized as follows:

(1) We design DeepObfuscator which is an adversarial training framework that can simultaneously defend against both reconstruction attack and private attribute leakage while maintaining the accuracy of primary learning tasks;
(2) We are the first to experimentally demonstrate that defending against only the reconstruction attack or private attribute leakage is *not* inclusive to each other;
(3) We quantitatively evaluate DeepObfuscator on CelebA and LFW datasets. The results show that the quality of reconstructed images from the obfuscated features is significantly decreased from 0.9458 to 0.3175 in terms of MS-SSIM, indicating that the person on the image is hardly reidentifiable visually. The classification accuracy of the inferred private attributes is reduced by around 30% to a random-guessing accuracy, but the accuracy of the intended classification tasks performed via the cloud service is reduced by only 2%.

(4) We demonstrate the efficiency of DeepObfuscator, showcasing real-time performance of the deployed models on smartphones.

The rest of this paper is organized as follows: Section 2 reviews the related work. Section 3 elaborates the design of DeepObfuscator. Section 4 evaluates the performance of DeepObfuscator. Section 5 shows two case studies of applying DeepObfuscator. Section 6 discusses the limitation of our proposed method. Section 7 concludes this work.

2 RELATED WORK

A large number of works have been done to protect data privacy using various anonymization techniques including k-anonymity [35], l-diversity [21] and t-closeness [16]. However, these solutions are designed for protecting sensitive attributes in a static database, and hence are not suitable to our addressed problem – obfuscating intermediate representations of data while retaining the utility for DNN inference. Differential privacy [1, 2, 4, 5, 31, 34, 37] is another widely applied technique to prevent an individual's data record from being leaked with a strong theoretical guarantee. But the privacy guarantee provided by differential privacy is different from the privacy protection offered by DeepObfuscator. The goal of differential privacy is to inject random noise to a user's data record such that an adversary cannot identify the existence of this data record in the database. Different from differential privacy, our goal is to hide private information from the intermediate representations, such that an adversary cannot accurately infer the protected private information and successfully reconstruct the raw data. Li *et al.* [15] present an information theoretic approach to hide private information in features while maximally retaining the information carried by the raw data, such that the extracted features still have high utility for training DNN models. However, defending against the reconstruction attack is not taken into account by this method. In addition, encryption-based approaches [7, 41] have been presented to protect data privacy, but they require to train specialized DNN models on the encrypted data. Unfortunately, such encryption-based solutions prevent general dataset release and introduce substantial computational overhead.

Osia *et al.* [25] combine dimensionality reduction, noise injection and Siamese fine-tuning to protect sensitive information from features, but it does not consider defending against the reconstruction attack. De-identification is another popular privacy-preserving method to prevent the identity from being visually recognized in many computer vision applications. There are various techniques to achieve de-identification, such as Gaussian blur [23], identity obfuscation [23], mean shift filtering [39] and adversarial image perturbation [24]. Although those approaches are effective in protecting visual privacy, they all degrade the utility of the data for DNN inference.

With recent developments of deep learning, several works have been proposed to protect data privacy by exploiting adversarial learning. Pittaluga *et al.* [30] propose an adversarial learning method to learn an encoder, aiming to defend against performing inference for specific attributes from the encoded intermediate representations. Seong *et al.* [24] design an adversarial network to transform the raw image so that the attacker cannot successfully perform image recognition. Wu *et al.* [40] present an adversarial framework to explicitly learn a degradation transform for the original video inputs in order to balance target task accuracy and the associated privacy budgets on the transformed video. Malekzadeh *et al.* [22] propose an on-device transformation of sensor data that will be used for specific applications, such as monitoring selected daily activities. This method can prevent the user from being identified by the adversary. Liu *et al.* propose PAN [17], which is an adversarial learning framework to obfuscate features for the privacy purpose. However, PAN defenses against the reconstruction attack by enlarging the difference between the raw image and the reconstructed image in terms of pixel-wise distance, which may not be able to guarantee a significant perceptual difference between the raw image and the reconstructed image (see Section 5.1). In addition, PAN's performance is evaluated only for protecting single private attribute, which is not always the case in practice.

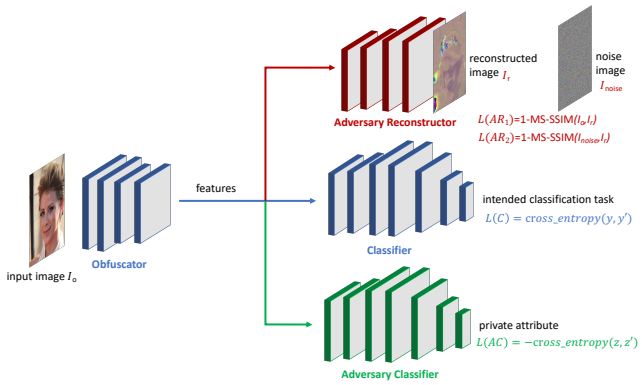


Figure 3: The design of DeepObfuscator.

3 DESIGN OF DEEPOBFUSCATOR

As Figure 3 shows, DeepObfuscator consists of three additional neural network modules: *classifier* (C), *adversary reconstructor* (AR) and *adversary classifier* (AC). The classifier works for the intended classification service. The adversary reconstructor and adversary classifier simulate an attacker in the adversarial training procedure, aiming to recover raw images and infer private attributes from the eavesdropped features. All the four modules are end-to-end trained using our proposed adversarial training algorithm.

Before presenting the details of each module, we give the following notations. We denote $D = \{I_1, \ldots, I_N\}$ as the images in the dataset, where N is the number of images, and $D' = \{I'_1, \ldots, I'_N\}$ represents the reconstructed images that are generated by the adversary reconstructor. Let $\mathcal{Y} = \{Y_1, \ldots, Y_M\}$ denote the set of the target classes that the classifier is trained to predict, and $Y_i = \{y_{i1}, \ldots, y_{iN}\}$ denotes the corresponding labels of each class. Similarly, we adopt $\mathcal{Z} = \{Z_1, \ldots, Z_K\}$ to denote the set of private classes that the adversary classifier aims to infer, and $Z_i = \{z_{i1}, \ldots, z_{iN}\}$ denotes the corresponding labels of each private class.

3.1 Obfuscator

The obfuscator (O) is a typical encoder which consists of an input layer, multiple convolutional layers, max-pooling layers and batch-normalization layers. The obfuscator is trained to hide privacy-related information while retaining useful information for intended classification tasks.

3.2 Classifier

The classifier (C) is jointly trained with the obfuscator as a complete CNN model. A service provider can choose any neural network architecture for the classifier based on task requirements and available computing resources. In DeepObfuscator, without loss of generality, we adopt a popular CNN architecture VGG16 [33], and split it into the obfuscator and the classifier.

The performance of the classifier C is measured using the cross-entropy loss function, which is expressed as:

$$\mathcal{L}(C) = -\sum_{j=1}^{\mathcal{N}}\sum_{i=1}^{\mathcal{M}} y_{ij} log(y'_{ij}) + (1 - y_{ij}) log(1 - y'_{ij}), \quad (1)$$

where $(y_{1j}, \ldots, y_{Mj})$ denote the ground truth labels for the jth data sample, and $(y'_{1j}, \ldots, y'_{Mj})$ are the corresponding predictions. Therefore, the obfuscator and the classifier can be optimized by minimizing the above loss function as:

$$\theta_o, \theta_c = \arg\min_{\theta_o, \theta_c} \mathcal{L}(C), \quad (2)$$

where θ_o and θ_c are the parameters of the obfuscator and classifier, respectively.

3.3 Adversary Classifier

By continuously querying the cloud service, an attacker can train the adversary classifier (AC) using the eavesdropped features as inputs and the interested private attributes as labels. An attacker can infer private attributes via feeding the eavesdropped features to the trained adversary classifier. In DeepObfuscator, we apply the same architecture to both the classifier and the adversary classifier. However, the attacker can choose any architecture for the adversary classifier. As we shall show in Section 4.4, the performance of using different architectures in both the classifier and the adversary classifier will not be significantly different from the one that is achieved using the same architecture.

Similar to the classifier, the performance of the adversary classifier AC is also measured using the cross-entropy loss function as:

$$\mathcal{L}(AC) = -\sum_{j=1}^{\mathcal{N}}\sum_{i=1}^{\mathcal{K}} z_{ij} log(z'_{ij}) + (1 - z_{ij}) log(1 - z'_{ij}), \quad (3)$$

where $(z_{1j}, \ldots, z_{Mj})$ denote the ground truth labels for the jth eavesdropped feature, and $(z'_{1j}, \ldots, z'_{Mj})$ stand for the corresponding predictions. When we simulate an attacker who tries to enhance the accuracy of the adversary classifier as high as possible, the adversary classifier needs to be optimized by minimizing the above loss function as:

$$\theta_{ac} = \arg\min_{\theta_{ac}} \mathcal{L}(AC), \quad (4)$$

where θ_{ac} is the parameter set of the adversary classifier. On the contrary, when defending against private attribute leakage, we train the obfuscator in our proposed adversarial training procedure that aims to degrade the performance of the adversary classifier while improving the accuracy of the classifier. Consequently, the obfuscator can be trained using Eq. 5 when simulating a defender:

$$\theta_o = \arg\min_{\theta_o} \mathcal{L}(C) - \lambda_1 \mathcal{L}(AC), \quad (5)$$

where λ_1 is a tradeoff parameter.

3.4 Adversary Reconstructor

The adversary reconstructor (AR), which is trained to recover the raw image from the eavesdropped features, also plays an attacker role. The attacker can apply any neural network architecture in the adversary reconstructor design. However, the worst case happens when an attacker knows the architecture of the obfuscator, and then builds the most powerful reconstructor, i.e., an exactly mirrored obfuscator by performing a layer-to-layer reversion. In DeepObfuscator, we adopt the most powerful reconstructor as the adversary reconstructor. The experiments in Section 4.4 show our trained obfuscator can successfully defend against the brute-force reconstruction attack when an attacker trains the reconstructor with different neural network architectures.

When playing as an attacker, the adversary reconstructor is trained to optimize the quality of the reconstructed image I_r as close as the original image I_o. In DeepObfuscator, we leverage MS-SSIM [20, 38] to evaluate the performance of the adversary reconstructor, which is expressed as:

$$\mathcal{L}(AR_1) = 1 - \text{MS-SSIM}(I_o, I_r). \quad (6)$$

The MS-SSIM value ranges between 0 and 1. The higher the MS-SSIM value is, the more perceptual similarity can be found between the two compared images, indicating a better quality of the reconstructed images. Consequently, an attacker can optimize the adversary reconstructor as:

$$\theta_{ar} = \arg\min_{\theta_{ar}} \mathcal{L}(AR_1), \quad (7)$$

where θ_{ar} is the parameter set of the adversary reconstructor. On the contrary, a defender expects to degrade the quality of the reconstructed image as much as possible. To this end, we generate one additional Gaussian noise image I_{noise}. The adversary reconstructor is trained to make each reconstructed image similar to I_{noise} but different from I_o, and the performance of the classifier should be maintained. When playing as a defender, the obfuscator can be trained as:

$$\mathcal{L}(AR_2) = 1 - \text{MS-SSIM}(I_{noise}, I_r) \quad (8)$$

$$\theta_o = \arg\min_{\theta_o} \mathcal{L}(C) + \lambda_2(\mathcal{L}(AR_2) - \mathcal{L}(AR_1)), \quad (9)$$

where λ_2 is a tradeoff parameter.

3.5 Adversarial Training Algorithm

Algorithm 1 summarizes the proposed four-stage adversarial training algorithm. Before performing the adversarial training, we first jointly train the obfuscator and the classifier without privacy concern to obtain the optimal performance on the intended classification tasks. Similarly, we also pre-train the adversary classifier and

Table 1: The architecture configurations of each module.

Obfuscator	Adversary Reconstructor	Classifier & Adversary Classifier
conv3-64	Upsample	3×conv3-256
conv3-64	deconv3-128	maxpool
maxpool	deconv3-64	3×conv3-512
conv3-128	Upsample	maxpool
conv3-128	deconv3-64	3×conv3-512
maxpool	deconv3-3	maxpool
		2×FC-4096
		FC-label length
		sigmoid

adversary reconstructor for initialization. As Algorithm 1 shows, within each epoch of training, each adversarial training iteration consists of four batches. In the first two batches, we train the obfuscator to defend against the adversary reconstructor and the adversary classifier while keeping the classifier unchanged. For the third batch, we optimize the adversary reconstructor and the adversary classifier by simulating an attacker, but the parameters of the obfuscator and the classifier are fixed. Finally, we optimize the classifier to improve the classification accuracy on the intended tasks.

Algorithm 1 Adversarial Training Algorithm

Input: Dataset $\mathcal{D}$
Output: $\theta_o, \theta_c, \theta_{ar}, \theta_{ac}$

Input: Dataset $\mathcal{D}$
for *every epoch* **do**
  for *every four batches* **do**
    if *batch idx mod* 4 == 0 **then**
      Defend against AR:
      $\mathcal{L}(C) + \mathcal{L}(AR_2) - \mathcal{L}(AR_1) \rightarrow$ update $O(\theta_o)$
    else if *batch idx mod* 4 == 1 **then**
      Defend against AC:
      $\mathcal{L}(C) - \mathcal{L}(AC) \rightarrow$ update $O(\theta_o)$
    else if *batch idx mod* 4 == 2 **then**
      reconstruction attack:
      $\mathcal{L}(AR_1) \rightarrow$ update $AR(\theta_{ar})$
      Infer private attributes:
      $\mathcal{L}(AC) \rightarrow$ update $AC(\theta_{ac})$
    else
      Recover C:
      $\mathcal{L}(C) \rightarrow$ update $C(\theta_c)$
    end if
  end for
end for

4 EVALUATION

In this section, we evaluate DeepObfuscator's performance on two real-world datasets, with a focus on the utility-privacy tradeoff. We also compare DeepObfuscator with existing solutions proposed in the literature and visualize the results.

4.1 Experiment Setup

We implement DeepObfuscator with PyTorch, and train it on a server with 4×NVIDIA TITAN RTX GPUs. We apply mini-batch technique in training with a batch size of 64, and adopt the AdamOptimizer [13] with an adaptive learning rate in all four stages in the adversarial training procedure. The architecture configurations of each module are presented in Table 1. We deploy the trained obfuscator on Google Pixel 2 and Pixel 3 to evaluate the real-time performance.

We adopt CelebA [19] and LFW [14] for the training and testing of DeepObfuscator. CelebA consists of more than 200K face images. Each face image is labeled with 40 binary facial attributes. The dataset is split into 160K images for training and 40K images for testing. LFW consists of more than 13K face images, and each face image is labeled with 16 binary facial attributes. We split LFW into 10K images for training and 3K images for testing.

4.2 Motivation

Before presenting our performance evaluations, we first verify our motivation that defending against only reconstruction attack or private attribute leakage is not inclusive to each other. We apply our proposed adversarial training algorithm to defend against only one of these two attacks each time, and evaluate the attack performance on the other. In this experiment, we select 'gender' as the private attribute. The results presented in Figure 2 verified our motivation.

One naïve solution of the exclusion of defending against reconstruction attack and private attribute leakage is to first train an obfuscator to defend against one of these two scenarios using the adversarial training approach, and then continue to train the obfuscator to defend against the other one. However, this naïve solution can not simultaneously defend against both the scenarios because the parameters of the obfuscator keep being updated in the second step. The above limitation motivates the design of DeepObfuscator.

4.3 Comparison Baselines

We select four types of data privacy-preserving baselines [17], which have been widely applied in the literature, and compare them with DeepObfuscator. The details settings of the baseline solutions are presented as below.

- **Noisy** method perturbs the raw data x by adding Gaussian noise $\mathcal{N}(0, \sigma^2)$, where σ is set to 40 according to [17]. The noisy data $\bar{x}$ will be delivered to the data collector. The Gaussian noise injected to the raw data can provide strong guarantees of differential privacy using less local noise. This scheme has been widely applied in federated learning [28, 36].
- **DP** approach injects Laplace noise the raw data x with diverse privacy budgets {0.1, 0.2, 0.5, 0.9}, which is a typical differential privacy method. The noisy data $\bar{x}$ will be submitted to the data collector.
- **Encoder** learns the latent representation of the raw data x using a DNN-based encoder. The extracted features z will be uploaded to the data collector.
- **Hybrid** method further perturbs the above encoded features by performing principle components analysis (PCA) and adding Laplace noise [25] with varying noise factors privacy budgets {0.1, 0.2, 0.5, 0.9}.

Table 2: Adversary reconstructor configurations.

URec#1	URec#2	ResRec
Input (54×44×128 feature maps)		
conv3-64 conv3-64 conv3-64	conv3-64 conv3-64	transconv3-64 $\begin{bmatrix} 3\times 3, & 64 \\ 3\times 3, & 64 \end{bmatrix} \times 2$
Upsample		
conv3-128 conv3-128 conv3-128	conv3-128 conv3-128	$\begin{bmatrix} 3\times 3, & 64 \\ 3\times 3, & 64 \end{bmatrix} \times 2$
Upsample		
conv3-256 conv3-256 conv3-256	conv3-256 conv3-256	conv1-3
conv3-3 conv1-3	conv3-3 conv1-3	sigmoid
sigmoid	sigmoid	

4.4 Effectiveness of Defending Against Reconstruction Attack

We quantitatively evaluate the quality of reconstructed images and obtain the results through a human perceptual study. Before showing quantitative results, we perform an experiment to simulate reconstruction attack using different reconstructor architectures. As introduced in Section 3.4, we adopt the most powerful decoder, i.e., exactly the reverse of the obfuscator, as the adversary reconstructor for training. However, an attacker may not be able to know the architecture of the obfuscator, and hence the attacker may conduct a brute-force attack using the reconstructor with different architectures. We implement three additional reconstructors as attackers in our experiments. The architectural configurations of those reconstructors are presented in Table 2. URec#1 and URec#2 are built based on the architecture of U-net [32], and ResRec is implemented with ResNet [11] architecture. Each reconstructor is separately trained with the same pre-trained obfuscator.

We again adopt the MS-SSIM to evaluate the quality of the reconstructed images that are generated by each reconstructor, i.e., comparing the similarity between the reconstructed image and the corresponding raw image. A smaller value of MS-SSIM implies less similarity between the reconstructed image and the raw image, indicating a more effective defense against reconstruction attacks. Table 3 presents the average MS-SSIM for attacking reconstructors on testing data. The results show that although we apply the mirrored obfuscator architecture for the adversary reconstructor when training the obfuscator, the trained obfuscator can effectively defend against reconstruction attacks no matter what kinds of architecture are adopted by an attacker in the reconstructor design.

Table 3: MS-SSIM for different attack reconstructors.

Training Reconstructor	Attack Reconstructor			
	AR in DeepObfuscator	URec#1	URec#2	ResRec
AR in DeepObfuscator	0.3175	0.3123	0.3095	0.3169

Quantitative Evaluation. In addition to MS-SSIM, we also adopt the Peak Signal to Noise Ratio (PSNR) – a widely used metric of image quality, to evaluate the quality of the reconstructed images. In this experiment, the obfuscator is trained by setting the intended classification task as 'smile' and the private attribute as 'gender' in CelebA. Smaller values of MS-SSIM and PSNR indicate a stronger defense against reconstruction attack. Table 4 presents the average MS-SSIM and PSNR on the testing data of DeepObfuscator and two baseline models. The result shows that DeepObfuscator is the most effective one to defend against reconstruction attack and the baseline models can hardly hide privacy information from the features. Figure 4 illustrates several examples of the reconstructed images. With our proposed adversarial training, the images that are reconstructed from the obfuscated features become unrecognizable. Even though directly applying Noisy method can hide more private information than apply Encoder method, the person in the image that is reconstructed from noisy features can still be re-identified. In summary, both quantitative evaluations and visual results show that DeepObfuscator can effectively defend against reconstruction attack.

Table 4: Average PSNR and MS-SSIM for DeepObfuscator and two baseline models.

Metric	DeepObfuscator	Encoder	Noisy
MS-SSIM	0.3175	0.9458	0.7263
PSNR	6.32	27.81	16.97

Human Perceptual Study. We also conduct an online human perceptual study to directly examine whether a person in a reconstructed image can be re-identified by humans. This study consists of 10 questions, each of which includes one reconstructed image and four raw images as options. One of the four options contains the person in the reconstructed image. Participants are instructed to choose the option that looks like the person the most in the reconstructed image. There is no time limit about how long the participants can see these images. Figure 5 shows one example question in the survey. It is very difficult to find hints from the reconstructed image to identify the correct answer, i.e., Figure 5(e). There are 40 participants involved in this survey and each participant can only submit one response, hence, we collect 40 responses in total in this study. The average re-identification accuracy of 10 questions is 28%, which is very close to a random guess, i.e., 25% for 4 options. Furthermore, we can imagine if there is no option offered to an attacker, it will become more challenging to re-identify the person from the reconstructed image alone.

4.5 Effectiveness of Defending Against Private Attribute Leakage

Comparison of utility-privacy tradeoff: We compare the utility-privacy tradeoff offered by DeepObfuscator with four privacy-preserving baselines. In our experiments, we set 'gender' and 'heavy makeup' as the private attributes to protect in CelebA, and consider detecting 'smile' and 'high cheekbone' as the intended classification tasks to evaluate the utility. With regard to LFW, we set 'gender' and 'Asian' as the private labels, and choose recognizing 'black

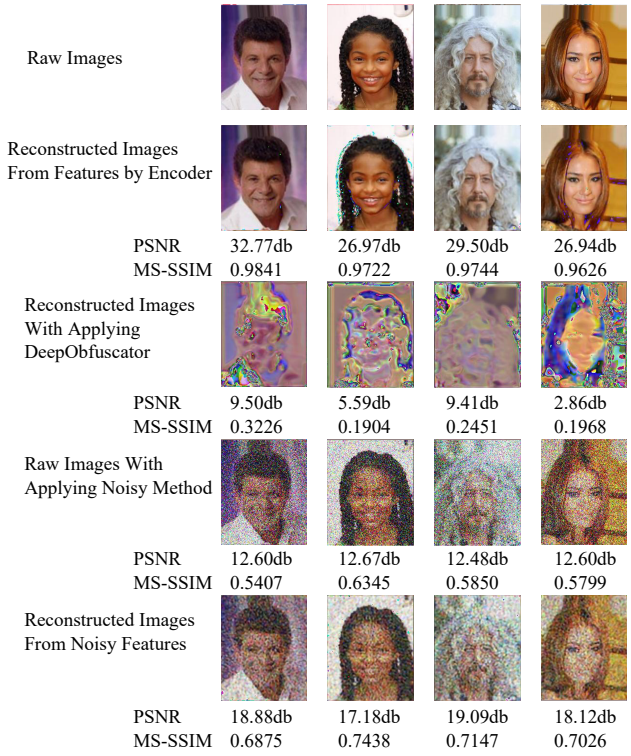


Figure 4: The comparison between reconstructed images with DeepObfuscator and two baseline models.

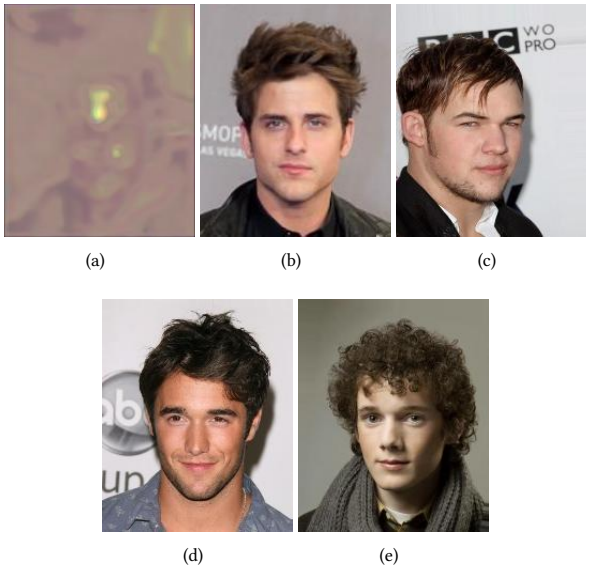


Figure 5: An example question of the human perceptual study. (a) is the reconstructed image, and (b)-(e) are the four options.

hair' and 'eyeglass' as the intended classification tasks. Figure 6 summarizes the utility-privacy tradeoff offered by four baselines and DeepObfuscator. Here we evaluate DeepObfuscator with four discrete choices of $\lambda_1 \in \{1, 2, 5, 10\}$ while setting $\lambda_2 = 1$. Note that the evaluation metric of privacy is defined as the classification accuracy of private attributes that we aim to protect, hence, the lower accuracy of private attributes indicates a better privacy protection. Therefore, the ideal case for the utility-privacy tradeoff should be in the lower right corner, which represents achieving the maximized utility while offering the strongest privacy protection.

As Figure 6 shows, although DeepObfuscator cannot always outperform the baselines in both utility and privacy, it still achieves the best utility-privacy tradeoff under most experiment settings. For example, in Figure 6(h), DeepObfuscator achieves the best tradeoff by setting $\lambda_1 = 1$. Specifically, the classification accuracy of 'Asian' on LFW is 52.37%, and the accuracy of 'eyeglass' is 84.47%. This demonstrates that DeepObfuscator can efficiently protect privacy while maintaining high accuracy of intended classification tasks.

In other four baselines, Encoder method can maintain best utility of the extracted features, but it fails to protect privacy due to the high accuracy of private attributes achieved by the attacker. Hybrid method can provide the most effective privacy protection, but the accuracy of intended classification tasks is unacceptable. In general, Noisy, DP and Hybrid methods offer strong privacy protection with sacrificing the utility.

Impact of the utility-privacy budget λ_1: An important step in the hybrid learning procedure is to determine the utility-privacy budget λ_1 and λ_2. Due to a large number of possible combinations of λ_1 and λ_2, we set $\lambda_2 = 1$ while varying λ_1 in this experiment. To determine the optimal λ_1, we evaluate the utility-privacy tradeoff on CelebA and LFW by setting different λ_1. Specifically, we evaluate the impact of λ_1 with four discrete choices of $\lambda \in \{1, 2, 5, 10\}$.

For experiments using CelebA, we choose detecting 'smile' and 'high cheekbone' as the intended classification tasks, and 'gender' and 'heavy makeup' as the private attributes that the attacker aims to infer from the obfuscated features. We design 6 testing sets using different combinations of those attributes: (1) {smile, gender}; (2) {high cheekbone, gender}; (3) {smile, high cheekbone, gender}; (4) {smile, gender, heavy makeup}; (5) {high cheekbone, gender, heavy makeup}; (6) {smile, high cheekbone, gender, heavy makeup}. In fact, those six testing sets can be divided into two groups: the first three sets only contain one private attribute, and the last three sets include two private attributes. Within each group, we explore how the different numbers of the intended classification tasks will affect the protection of the private attributes. In addition, if we compare each testing set between two groups accordingly, we can investigate how the accuracy of the intended tasks will change with the number of the private attributes that need to be protected.

Figure 7 shows the average accuracy of intended tasks and private attributes using the classifier and adversary classifier which are trained in the way adopted by DeepObfuscator. With the proposed adversarial training, DeepObfuscator can effectively prevent private attributes from being inferred by an attacker while only incurring a small accuracy drop on the intended classification tasks. In general, the larger λ_1 can provides a stronger privacy protection

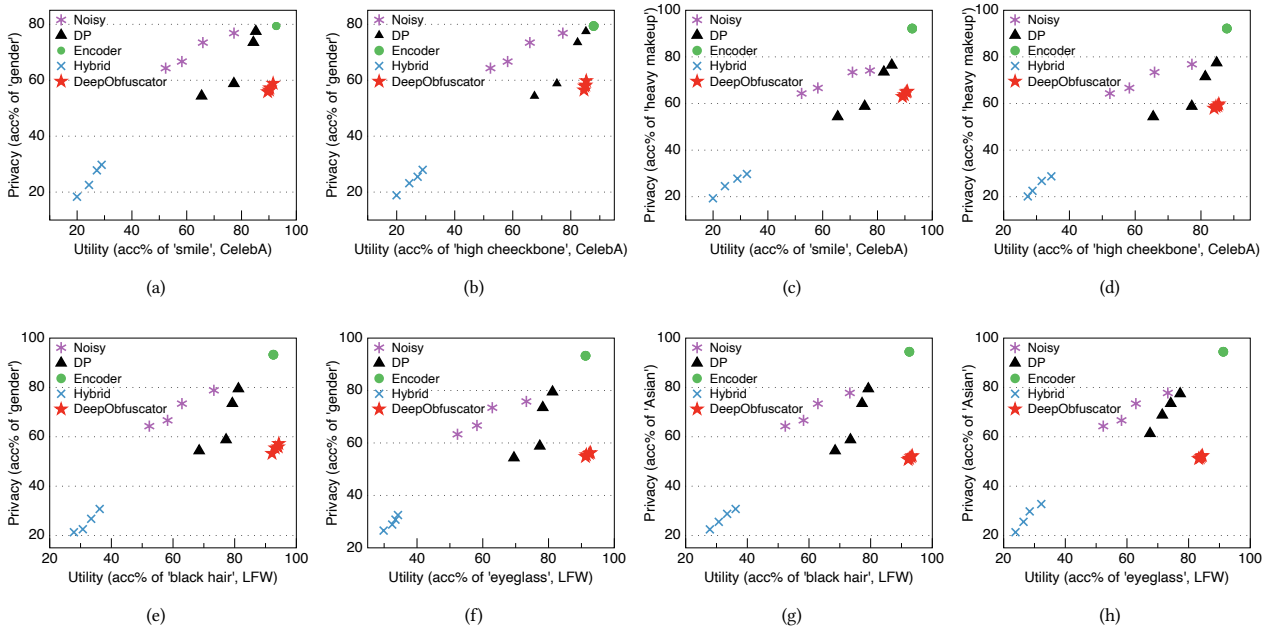


Figure 6: Utility-privacy tradeoff comparison of DeepObfuscator with four baselines on CelebA and LFW. The higher the utility score, the better. The lower the privacy score, the better. Therefore, the best performing algorithms are those located towards the bottom right quadrant.

(i.e., lower accuracy of private attributes), but degrades the performance on intended classification tasks. For example, as Figure 7 shows, the accuracy of 'gender' decreases from 58.85% to 55.85% and the accuracy of 'smile' decreases from 91.53% to 89.52%, when increasing λ_1 from 1 to 10. However, with the increasing λ_1, the performance drop of both private attributes and intended classification tasks will be marginal.

If we compare Figure 7(c) vs. Figure 7(a-b) and Figure 7(f) vs. Figure 7(d-e), given a particular λ_1, it can be observed that performing more intended tasks will weaken the defense against private attribute leakage due to the intrinsic correlation between the intended tasks and the private attributes that an attacker aims to infer. Specifically, the accuracy of 'gender' slightly increases from 58.85% in Figure 7(a) and 59.79% in Figure 7(b) to 63.96% in Figure 7(c) when setting $\lambda_1 = 1$. Similarly, the accuracy of 'gender' and 'heavy makeup' increase by 1% from Figure 7(d-e) to Figure 7(f). In addition, by comparing Figure 7(a-c) vs. Figure 7(d-f), we found that protecting more private attributes leads to slight decrease in the accuracy of the intended tasks under a specific λ_1. For example, the accuracy of 'smile' slightly decreases from 91.53% in Figure 7(a) to 90.89% in Figure 7(d) when setting $\lambda = 1$. The reason is that the feature related to the private attributes has some intrinsic correlations to the feature related to the intended tasks. Therefore, more correlated features may be hidden if more private attributes need to be protected. As a result, the performance of the intended tasks becomes harder to maintain.

Similar to the above experiments on CelebA, in LFW, we choose recognizing 'eyeglass' and 'black hair' as the intended classification tasks, and 'gender' and 'Asian' as the private attributes that the attacker aims to infer from the obfuscated features. We also design 6 testing sets using different combinations of those attributes: (1) {eyeglass, gender}; (2) {black hair, gender}; (3) {eyeglass, black hair, gender}; (4) {eyeglass, gender, Asian}; (5) {black hair, gender, Asian}; (6) {eyeglass, black hair, gender, Asian}. As same as the settings for CelebA, those six testing sets can be divided into two groups: the first three sets only contain one private attribute, and the last three sets include two private attributes. We also conduct the same intra-group and inter-group comparisons as the above evaluations on CelebA.

Generally, we observed the result as same as that of CelebA - performing more intended tasks will weaken the defense against private attribute leakage given a specific λ_1. For example, if we compare Figure 8(c) vs. Figure 8(a-b), the accuracy of 'gender' slightly increases from 56.27% in Figure 8(a) and 57.23% in Figure 8(b) to 59.74% in Figure 8(c) with $\lambda_1 = 1$. Similarly, the accuracy of 'gender' and 'Asian' increase by 1% from Figure 8(d-e) to Figure 8(f). In addition, by comparing Figure 8(a-c) vs. Figure 8(d-f), we also found that protecting more private attributes leads to slight decrease in the accuracy of the intended tasks. For example, the accuracy of 'eyeglass' slightly decreases from 92.78% in Figure 8(a) to 91.25% in Figure 8(d) when setting $\lambda_1 = 1$.

Cross-Dataset Evaluation. We also conduct cross-dataset evaluations by training the obfuscator using either CelebA or LFW dataset and test the performance on the other dataset. Specifically, we choose recognizing 'black hair' as the intended classification task, and 'gender' as the private attribute that the attacker aims to infer from the obfuscated features. As Table 5 illustrates, the obfuscator that is trained using one dataset can still effectively defend

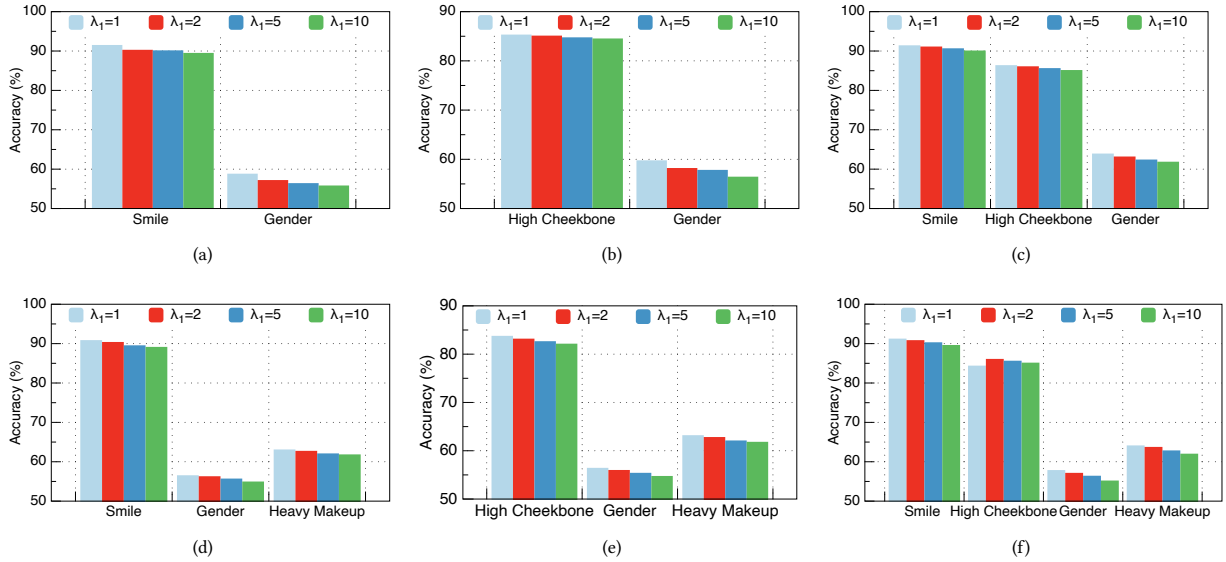


Figure 7: The impact of the utility-privacy budget λ_1 on CelebA. ($\lambda_2 = 1$)

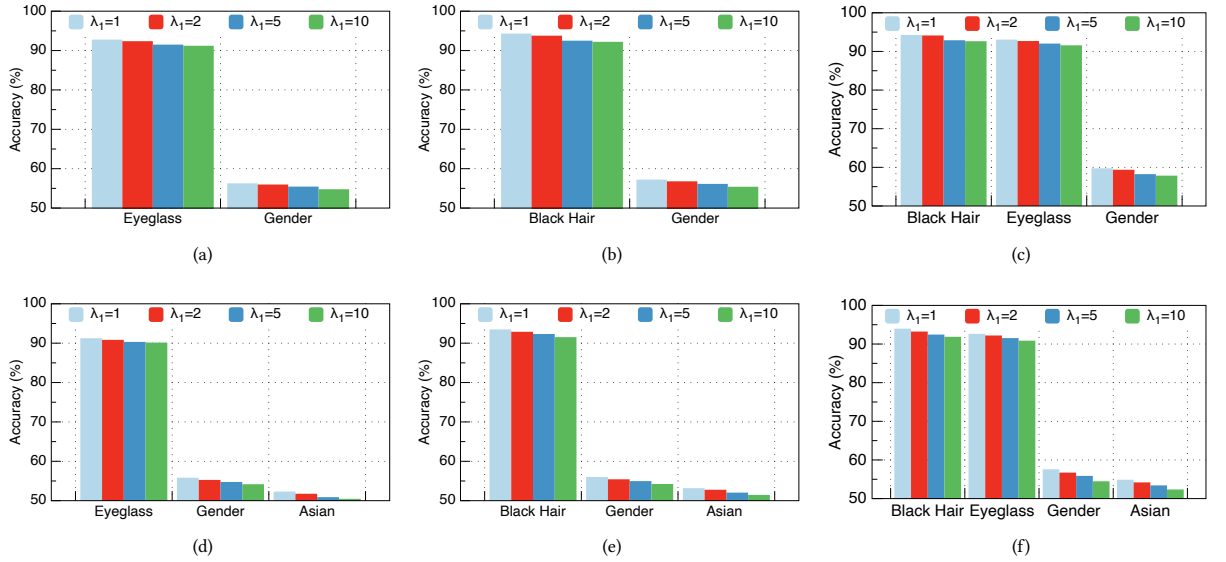


Figure 8: The impact of the utility-privacy budget λ_1 on LFW. ($\lambda_2 = 1$).

against private attribute leakage on the other dataset, while maintaining the classification accuracy of the the intended classification task. For example, if we train the obfuscator using CelebA and then test it on LFW, the accuracy of 'gender' decreases to 53.74% compared with 57.23% by directly training the obfuscator using LFW. The accuracy of 'black hair' marginally increases to 94.79% from 94.31%. The reason is that CelebA offers a larger number of training data so that the obfuscator can be trained for a better performance. Although there is a marginal performance drop, the obfuscator that is trained using LFW still works well on CelebA. The cross-dataset evaluations demonstrate the transferability of DeepObfuscator.

4.6 Performance on Smartphones

We evaluate the real-time performance of deploying the trained obfuscator on Google Pixel 2 and Pixel 3, including latency, storage and energy consumption. We randomly select 1000 images from CelebA, and feed them into the trained obfuscator which is deployed on smartphones. The averaged results are summarized in Table 6.

Table 5: Evaluate the transferability of DeepObfuscator with cross-dataset experiments.

Test Dataset	Training Dataset	'gender'	'black hair'
LFW	CelebA	53.74%	94.79%
LFW	LFW	57.23%	94.31%
CelebA	LFW	59.87%	93.57%
CelebA	CelebA	58.82%	94.88%

Table 6: Performance of running the learned obfuscator on Google Pixel 2 and Pixel 3.

Smartphone	Latency (ms)	Storage (MB)	Energy (mJ)
Google Pixel 2	105	5.6	2.8
Google Pixel 3	101	5.6	2.7

The learned obfuscator only occupies 5.6 MB of memory, costs 101-105 ms for extracting features, and consumes 2.7-2.8 mJ of energy for each feature extraction pass.

5 CASE STUDIES

5.1 Case Study on Driver Behavior Recognition

Besides facial recognition tasks which are binary classifications, performing evaluations on multi-class tasks are essential to verify DeepObfuscator's performance. Therefore, we also evaluate DeepObfuscator on StateFarm dataset [12] that contains 22424 images (17939 for training, 4485 for testing) of 10 different driver behaviors from 26 people. In this experiment, driver behavior recognition is considered as an intended task while the driver's identity is a private attribute we want to protect. We consider the Encoder presented in Section 4.1 as the compared baseline. With our adversarial training method, the accuracy of driver behavior classification slightly drops from 98.32% to 95.49%, but the accuracy of driver identity recognition decreases significantly from 99.97% to 30.38%. Figure 9 shows the images reconstructed using the features encoded by DeepObfuscator, indicating that DeepObfuscator still successfully defend against the attacker's reconstruction attack.

5.2 Case Study on Text Data

DeepObfuscator can be easily extended to many other applications with various modalities of data. For example, we replace the CNN architecture with LSTM-based architecture in DeepObfuscator, and then evaluate performance on TwitterAAE dataset described in [3]. TWitterAAE consists of 166K and 10K tweets for training and testing, respectively. In our experiment, binary mention detection is considered as an intended task while the race (AAE (African-American English) or SAE (Standard American English)) is a binary private attribute we aim to protect. The mention detection is a binary classification task to determine if a tweet mentions another user, i.e, classifying conversational vs. non-conversational tweets. The architecture configurations are presented in Table 7. With applying DeepObfuscator, the mention detection accuracy marginally drops from 81.69% to 81.38%, however, the accuracy of race classification decreases from 79.86% to 52.78%, which is a totally random-guessing level.

Table 7: The architecture configurations of LSTM-based module.

Obfuscator	Classifier & Adversary Classifier
Embedding-300	2×LSTM-300
LSTM-300	FC-150
	ReLU
	FC-label length

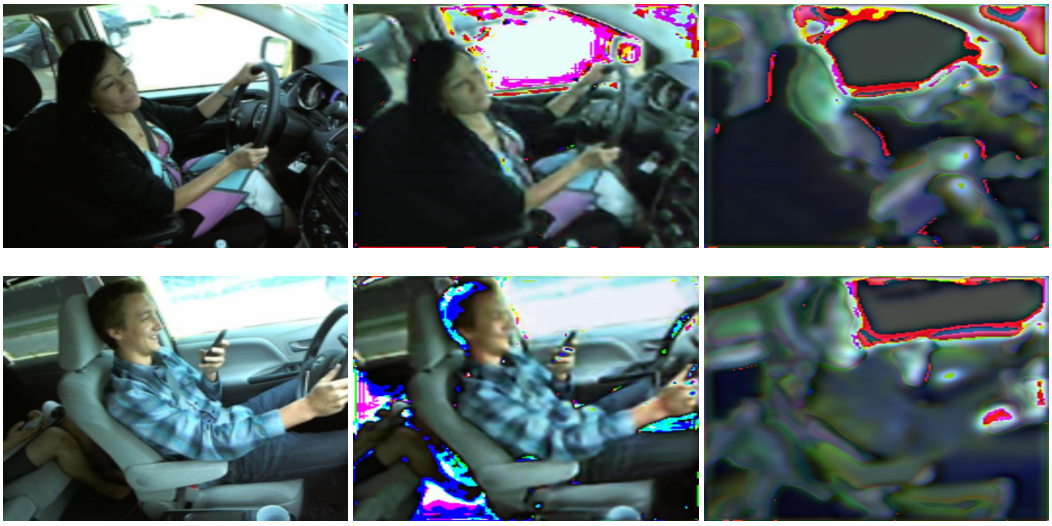
5.3 Comparison with PAN

PAN [17] also provides privacy protection against the reconstruction attack by introducing a loss function for the adversary reconstructor, and the loss function is based on the Euclidean distance between a raw image and a reconstructed image. In the adversarial training, PAN aims to enlarge the difference between the raw image and the reconstructed image in terms of pixel-wise distance. In contrast to PAN, we adopt MS-SSIM as the metric to evaluate the perceptual similarity between the raw image and the reconstructed image in the loss function. More important, DeepObfuscator makes each reconstructed image similar to a crafted Gaussian noise image which is utilized in training, but significantly different from the raw image in terms of MS-SSIM. Here, we compare the effectiveness of defending against the reconstruction attack between DeepObfuscator and PAN via visualizing the reconstructed images. Specifically, we replace our loss function with PAN's Euclidean distance based loss function for comparisons, but keep anything else unchanged. As Figure 10 shows, with applying the PAN's loss function, several pixels around the face are significantly changed in the reconstructed images compared with the raw images. However, the key features of the person in the reconstructed images can still be easily identified. On the contrary, it is very difficult to identify distinguishable information from the reconstructed images when applying DeepObfuscator.

6 DISCUSSION

We evaluate the real-time performance of running the learned obfuscator on smartphones, and results show the applicability of our proposed method. However, the efficiency is a major concern about deploying the learned obfuscator on resource-constrained smartphones and edge devices. We have not perform the model optimization for the obfuscator in terms of efficiency. Numerous works have been done for model compression, such as quantization [8], pruning [9], etc. We propose to apply those approaches to optimize the efficiency of running the learned obfuscator on smartphone.

We evaluate DeepObfuscator on image and text datasets, but it can be easily extended to many other applications. For example, if we replace the CNN architecture with a task-specific recurrent neural network, it is also feasible to apply DeepObfuscator to other data modalities, such as sensor data (e.g., accelerometer, gyroscope). We will evaluate DeepObfuscator using various formats of data in the future work.



(a) raw images (b) reconstructed images with applying Encoder (c) reconstructed images with applying DeepObfuscator

Figure 9: Reconstructed images of the driver behavior recognition task.

(a) raw image (b) reconstructed image using PAN's loss function (c) reconstructed image using DeepObfuscator

Figure 10: Examples of reconstructed images using DeepObfuscator and PAN.

Even though DeepObfuscator attains a notably better privacy-utility tradeoff than existing works, it requires the prior knowledge of primary learning tasks before training. If the primary learning tasks are changed, we need to retrain the DeepObfuscator from scratch to achieve a good privacy-utility tradeoff. However, such requirement may limit the applicability and generalization of DeepObfuscator in practice. We plan to augment DeepObfuscator with information theory-based method such that that the learned feature extractor can hide the privacy information from the intermediate representations; while maximally retaining the original information embedded in the raw data.

7 CONCLUSION

We proposed an adversarial training framework DeepObfuscator for privacy-preserving image classifications by simultaneously defending against both reconstruction attack and private attribute leakage. DeepObfuscator consists of an obfuscator, a classifier, an adversary reconstructor and an adversary classifier. The obfuscator is trained using our proposed end-to-end adversarial training algorithm to hide sensitive information which can be exploited to reconstruct raw images and infer private attributes by an attacker. Useful features for the intended classification tasks are still retained by the obfuscator. The adversary reconstructor and adversary classifier play an attacker role in the adversarial training procedure, aiming to reconstruct the raw image and infer private attributes from the eavesdropped features. Evaluations on CelebA and LFW datasets show that the quality of the reconstructed images from the obfuscated features is significantly decreased from 0.9458 to 0.3175 in terms of MS-SSIM, indicating the person on the reconstructed images is hardly to be re-identified. The classification accuracy of the inferred private attributes that can be achieved by the attacker significantly drops down to a random-guessing level, but the accuracy of the intended classification tasks performed via the cloud service drops by mere 2%. The cross-dataset evaluations demonstrate the transferability of DeepObfuscator, indicating a great practicability in the real world.

REFERENCES

[1] Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. 2017. {BLENDER}: Enabling local search with a hybrid differential privacy model. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 747–764.

[2] Raef Bassily and Adam Smith. 2015. Local, private, efficient protocols for succinct histograms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 127–135.

[3] Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic Dialectal Variation in Social Media: A Case Study of African-American English.

In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 1119–1130. https://doi.org/10.18653/v1/D16-1120

[4] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2013. Local privacy and statistical minimax rates. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 429–438.

[5] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[6] Clément Feutry, Pablo Piantanida, Yoshua Bengio, and Pierre Duhamel. 2018. Learning anonymized representations with adversarial neural networks. *arXiv preprint arXiv:1802.09386* (2018).

[7] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*. 201–210.

[8] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning*. 1737–1746.

[9] Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149* (2015).

[10] Jianping He and Lin Cai. 2017. Differential private noise adding mechanism: Basic conditions and its application. In *2017 American Control Conference (ACC)*. IEEE, 1673–1678.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[12] Kaggle. 2019. State Farm Distracted Driver Detection. https://www.kaggle.com/c/state-farm-distracted-driver-detection.

[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv.org* (Dec. 2014), arXiv:1412.6980. arXiv:cs.LG/1412.6980

[14] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. 2009. Attribute and simile classifiers for face verification. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 365–372.

[15] Ang Li, Yixiao Duan, Huanrui Yang, Yiran Chen, and Jianlei Yang. 2020. TIPRDC: task-independent privacy-respecting data crowdsourcing framework for deep learning with anonymized intermediate representations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 824–832.

[16] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 106–115.

[17] Sicong Liu, Junzhao Du, Anshumali Shrivastava, and Lin Zhong. 2019. Privacy Adversarial Network: Representation Learning for Mobile Data Privacy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 4 (2019), 1–18.

[18] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. 2015. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE international conference on computer vision*. 1377–1385.

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[20] Kede Ma, Qingbo Wu, Zhou Wang, Zhengfang Duanmu, Hongwei Yong, Hongliang Li, and Lei Zhang. 2016. Group mad competition-a new methodology to compare objective image quality models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1664–1673.

[21] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5188–5196.

[22] Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. 2019. Mobile Sensor Data Anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation* (Montreal, Quebec, Canada) *(IoTDI '19)*. Association for Computing Machinery, New York, NY, USA, 49–58. https://doi.org/10.1145/3302505.3310068

[23] Seong Joon Oh, Rodrigo Benenson, Mario Fritz, and Bernt Schiele. 2016. Faceless person recognition: Privacy implications in social media. In *European Conference on Computer Vision*. Springer, 19–35.

[24] Seong Joon Oh, Mario Fritz, and Bernt Schiele. 2017. Adversarial image perturbation for privacy protection a game theory perspective. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 1491–1500.

[25] Seyed Ali Osia, Ali Shahin Shamsabadi, Sina Sajadmanesh, Ali Taheri, Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. 2020. A hybrid deep learning architecture for privacy-preserving mobile analytics. *IEEE Internet of Things Journal* (2020).

[26] Seyed Ali Osia, Ali Shahin Shamsabadi, Ali Taheri, Kleomenis Katevas, Hamid R Rabiee, Nicholas D Lane, and Hamed Haddadi. 2017. Privacy-preserving deep inference for rich user data on the cloud. *arXiv preprint arXiv:1710.01727* (2017).

[27] Seyed Ali Osia, Ali Taheri, Ali Shahin Shamsabadi, Minos Katevas, Hamed Haddadi, and Hamid RR Rabiee. 2018. Deep private-feature extraction. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[28] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908* (2018).

[29] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. 2015. Deep face recognition.. In *bmvc*, Vol. 1. 6.

[30] Francesco Pittaluga, Sanjeev Koppal, and Ayan Chakrabarti. 2019. Learning privacy preserving encodings through adversarial training. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 791–799.

[31] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 192–203.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[33] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[34] Adam Smith, Abhradeep Thakurta, and Jalaj Upadhyay. 2017. Is interaction necessary for distributed private learning?. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 58–77.

[35] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[36] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. 2019. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*. 1–11.

[37] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally differentially private protocols for frequency estimation. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 729–745.

[38] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. 2003. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, Vol. 2. Ieee, 1398–1402.

[39] Thomas Winkler, Adám Erdélyi, and Bernhard Rinner. 2014. TrustEYE. M4: protecting the sensor—not the camera. In *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 159–164.

[40] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. 2018. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 606–624.

[41] Ryo Yonetani, Vishnu Naresh Boddeti, Kris M Kitani, and Yoichi Sato. 2017. Privacy-preserving visual learning using doubly permuted homomorphic encryption. In *Proceedings of the IEEE International Conference on Computer Vision*. 2040–2050.