DeepObfuscator

DeepObfuscator

Obfuscating Intermediate Representations with Privacy-Preserving Adversarial Learning on Smartphones

- Ang Li – Duke University
- Jiayi Guo – Tsinghua University
- Huanrui Yang – Duke University
- Flora D. Salim – RMIT University
- Yiran Chen – Duke University

Obfuscator?

obfuscator noun

Save Word

ob·fus·ca·tor | \ äbˈfəˌskātə(r), əbˈ-; ˈäb(ˌ)fəˌ- \

plural -s

Definition of *obfuscator*

: one that obfuscates

Obfuscate?

obfuscate verb

Save Word

ob·fus·cate | \ ˈäb-fə-ˌskāt ; äb-ˈfə-ˌskāt, əb- \

obfuscated; **obfuscating**

Definition of *obfuscate*

transitive verb

1 **a** **:** to throw into shadow **:** DARKEN

b **:** to make obscure

// *obfuscate* the issue

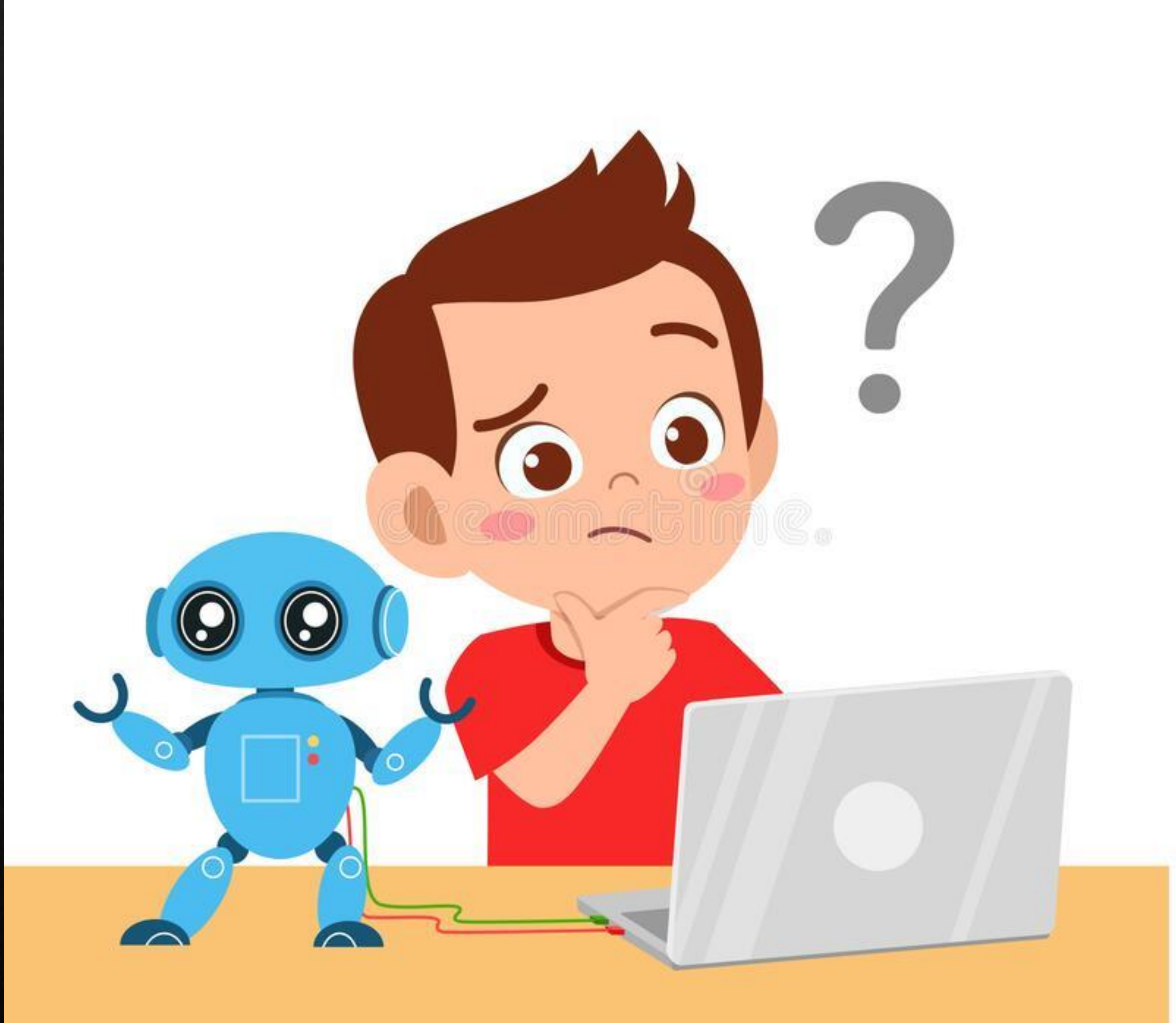
// officials who ... continue to obscure and *obfuscate* what happened
— Mary Carroll

2 **:** CONFUSE

// *obfuscate* the reader

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 maintain useful features for image classifications.

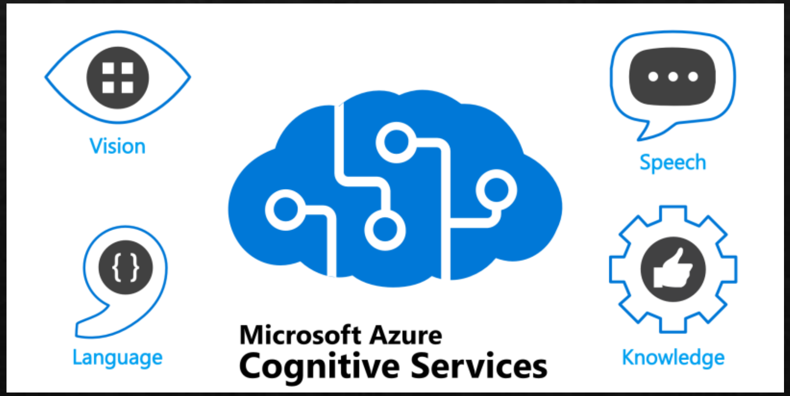


Why?

- Hard to run deep learning models on mobile devices
  - Computational resource limitations
- Alternative solution
  - The cloud

Why?

- Large-sized deep-learning-based applications are deployed on cloud servers (MLaaS)
  - Amazon Rekognition
  - Microsoft Cognitive Services
- Require users to send data (images) to cloud provider
- Leads to privacy concern due to vulnerable data
  - Age
  - Gender



Propositions

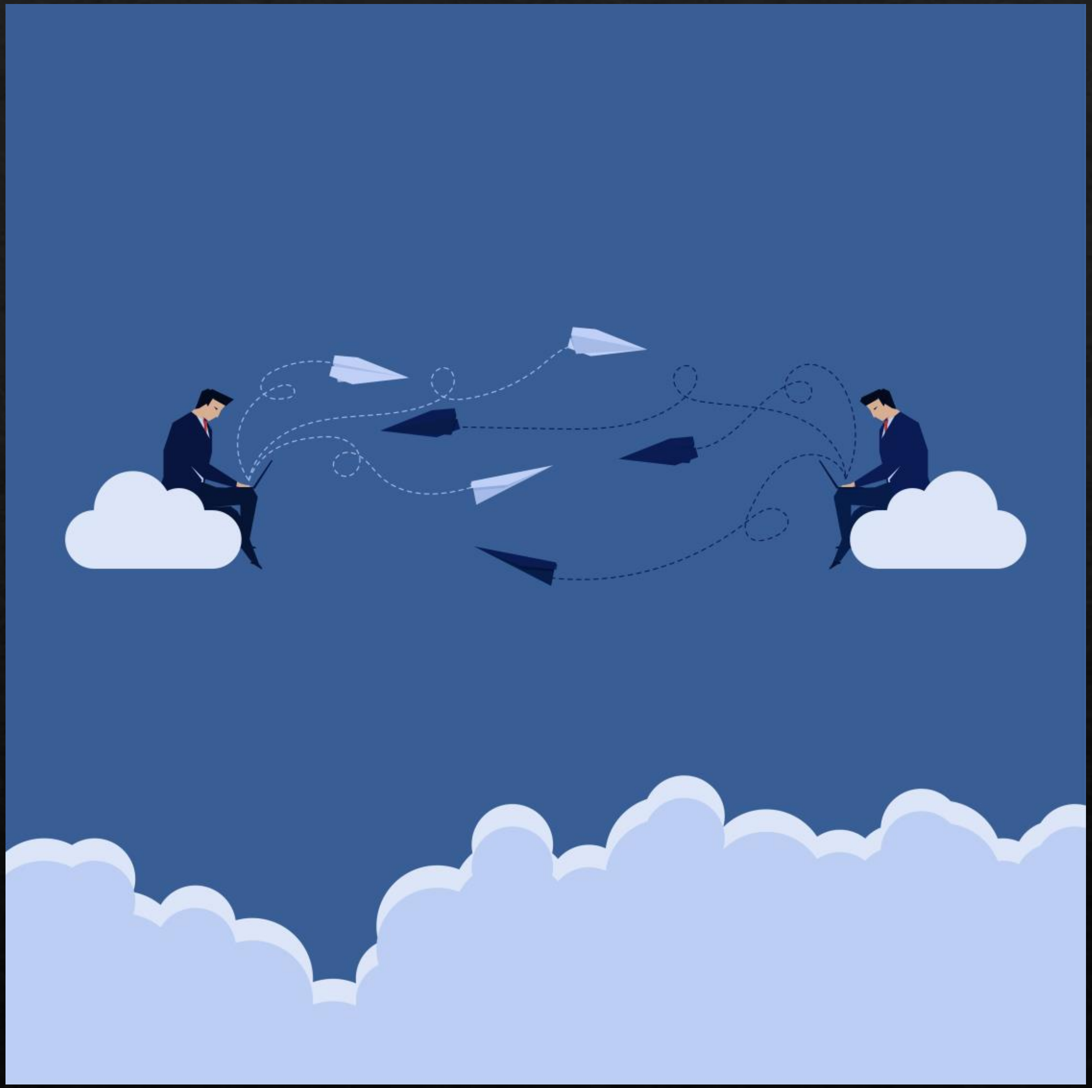
- Send features extracted from raw data to the cloud
  - Can still be exploited by attackers to recover raw images and to infer private attributes like age and gender

DeepObfuscator: Overview

The algorithm simulates the game between an attacker who makes efforts to reconstruct raw image and infer private attributes from the extracted features who aims to protect user privacy

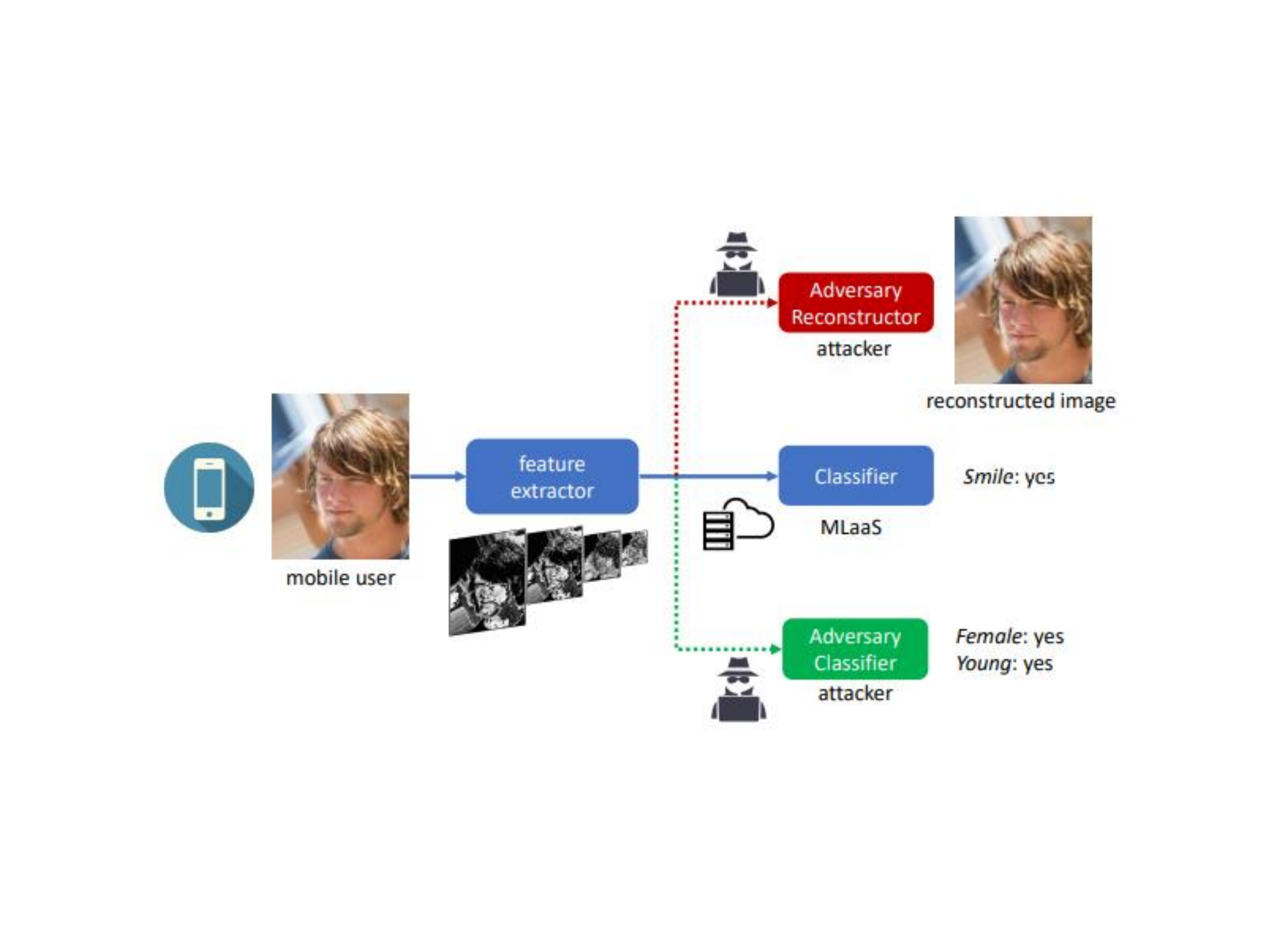
Can deploy the trained obfuscator on the smart phone

Features can be locally extracted and then sent to the cloud



Reconstruction Attacks

An example where the reconstruction attack and private attribute leakage occur in a MLaaS for facial attribute recognition

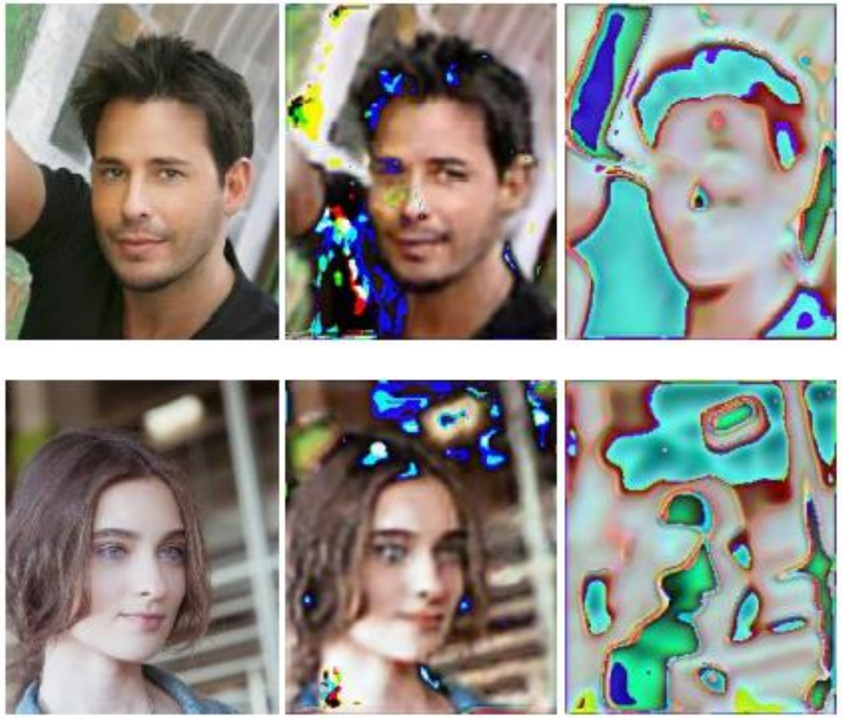


Reconstructed Images

a) Raw images

b) Image reconstructed, defending against only private attribute leakage

c) Image reconstructed, defending against only reconstruction attack



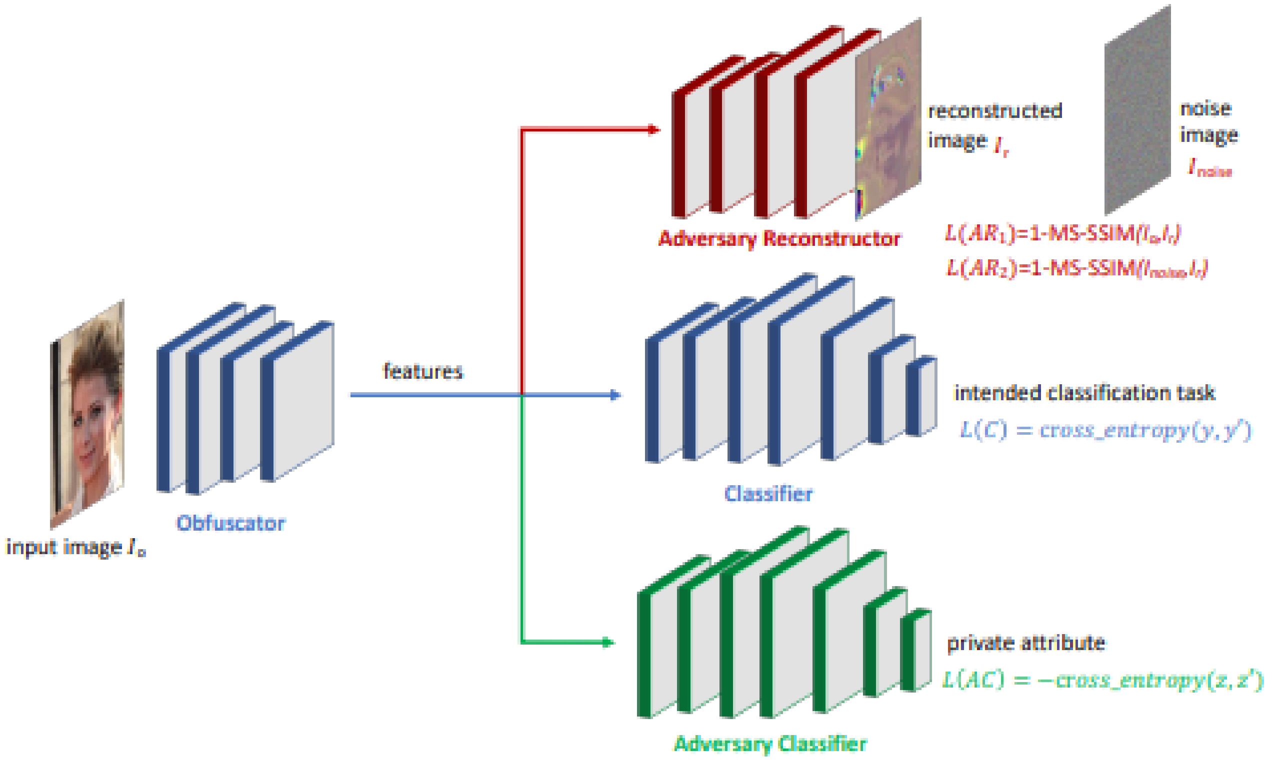
(a) (b) (c)

Defense against Reconstruction Attacks: Related Works

Anonymization techniques

- Designed for protecting sensitive attributes in a static database
- Not suitable for obfuscating intermediate representations of data while retaining the utility for deep neural network inference.

Design



reconstructed image I_r
noise image I_{noise}
Adversary Reconstructor
$L(AR_1)$=1-MS-SSIM(I_o,I_r)
$L(AR_2)$=1-MS-SSIM(I_{noise},I_r)
features
intended classification task
$L(C) = cross\_entropy(y, y')$
Classifier
input image I_o
Obfuscator
private attribute
$L(AC) = -cross\_entropy(z, z')$
Adversary Classifier

Obfuscator

The obfuscator is a typical encoder which consists of an input layer, multiple convolutional layers, max-pooling layers and batch-normalization layers.

Trained to hide privacy-related information while retaining useful information for intended classification tasks

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.

DeepObfuscator adopts VGG16

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

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

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:

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

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

Adversary Classifier

By continuously querying the cloud service, an attacker (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

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

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

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:

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

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:

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

where λ_1 is a tradeoff parameter.

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.

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:

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

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:

$$\theta_{ar} = \underset{\theta_{ar}}{\arg\min} \mathcal{L}(AR_1), \tag{7}$$

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:

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

$$\theta_o = \underset{\theta_o}{\arg\min} \mathcal{L}(C) + \lambda_2(\mathcal{L}(AR_2) - \mathcal{L}(AR_1)), \tag{9}$$

where λ_2 is a tradeoff parameter.

Adversarial Training Algorithm

First: jointly train obfuscator and classifier without privacy concern to obtain the optimal performance on the intended classification tasks. Also pre-train adversary classifier and adversary reconstructor for initialization.

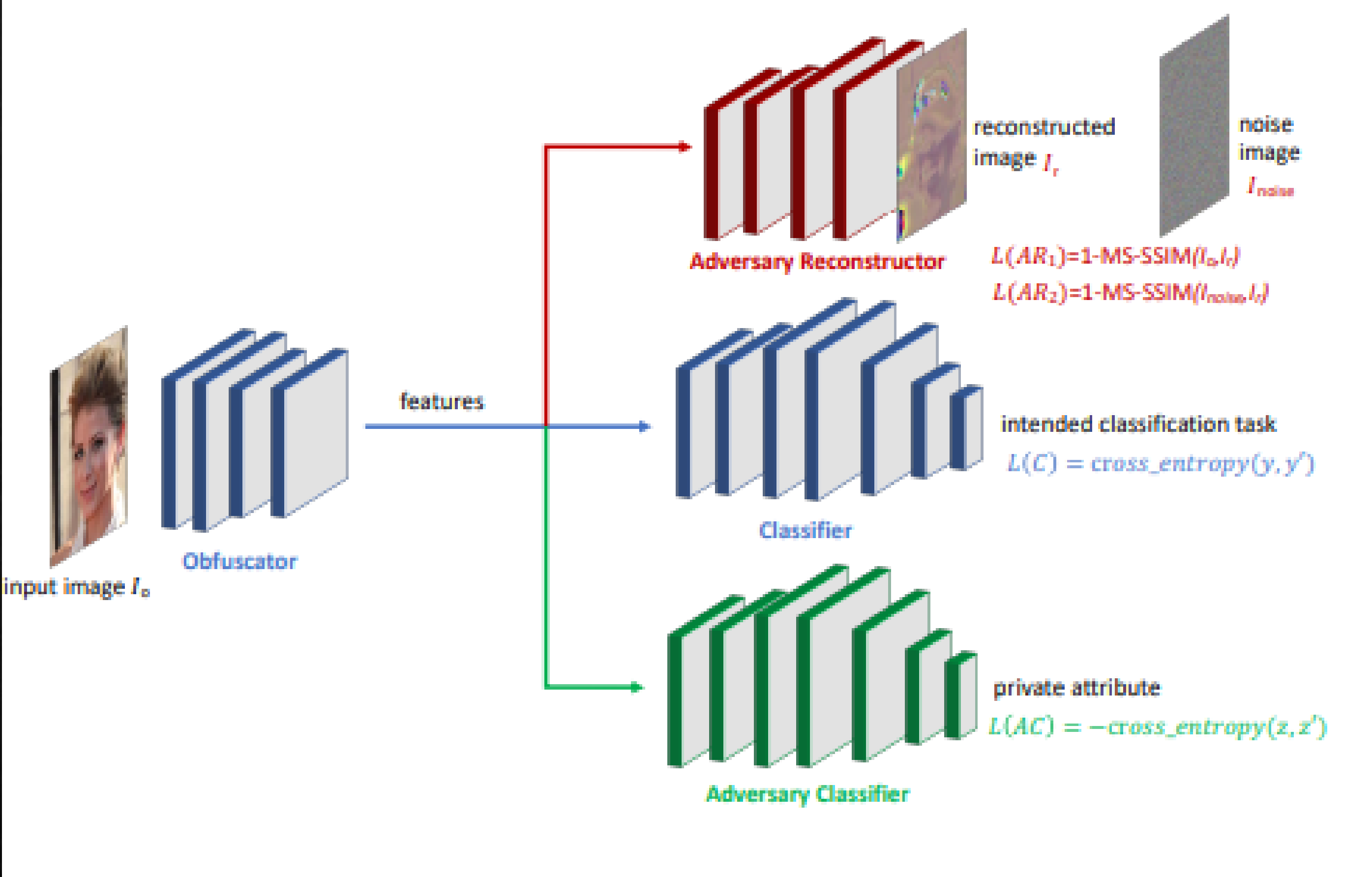
Algorithm 1 Adversarial Training Algorithm

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

Technical Objective

Apply adversarial training for maximizing the reconstruction error of the adversary reconstructor and the classification error of the adversary classifier, but minimize the classification error of the intended classifier



Experiment Setup

- Implemented with PyTorch
- Trained on server with 4xNVIDIA TITAN RTX GPUs
- Adopt AdamOptimizer with an adaptive learning rate

Experiment Setup: Datasets

- CelebA
  - 200k face images
  - 40 binary facial attributes
  - 160k images for training
  - 40k images for testing
- LFW
  - 13k face images
  - 16 binary facial attributes
  - 10k images for training
  - 3k images for testing

Evaluation

Evaluate DeepObfuscator's performance on two real-world datasets, with a focus on the utility-privacy tradeoff.

Also compare DeepObfuscator with existing solutions proposed in the literature and visualize the results.

Comparison Baselines

- **Noisy method** perturbs the raw data x by adding Gaussian noise N (0, σ^2), where σ is set to 40
- **DP** approach injects Laplace noise the raw data x with diverse privacy budgets {0.1, 0.2, 0.5, 0.9}
- **Encoder** learns the latent representation of the raw data x using a DNN-based encoder
- **Hybrid method** further perturbs the above encoded features by performing principle components analysis (PCA) and adding Laplace noise with varying privacy budgets

Effectiveness of Defending Against Reconstruction Attack

- MS-SSIM is used to evaluate quality of reconstructed images
- Smaller value of MS-SSIM implies less similarity

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

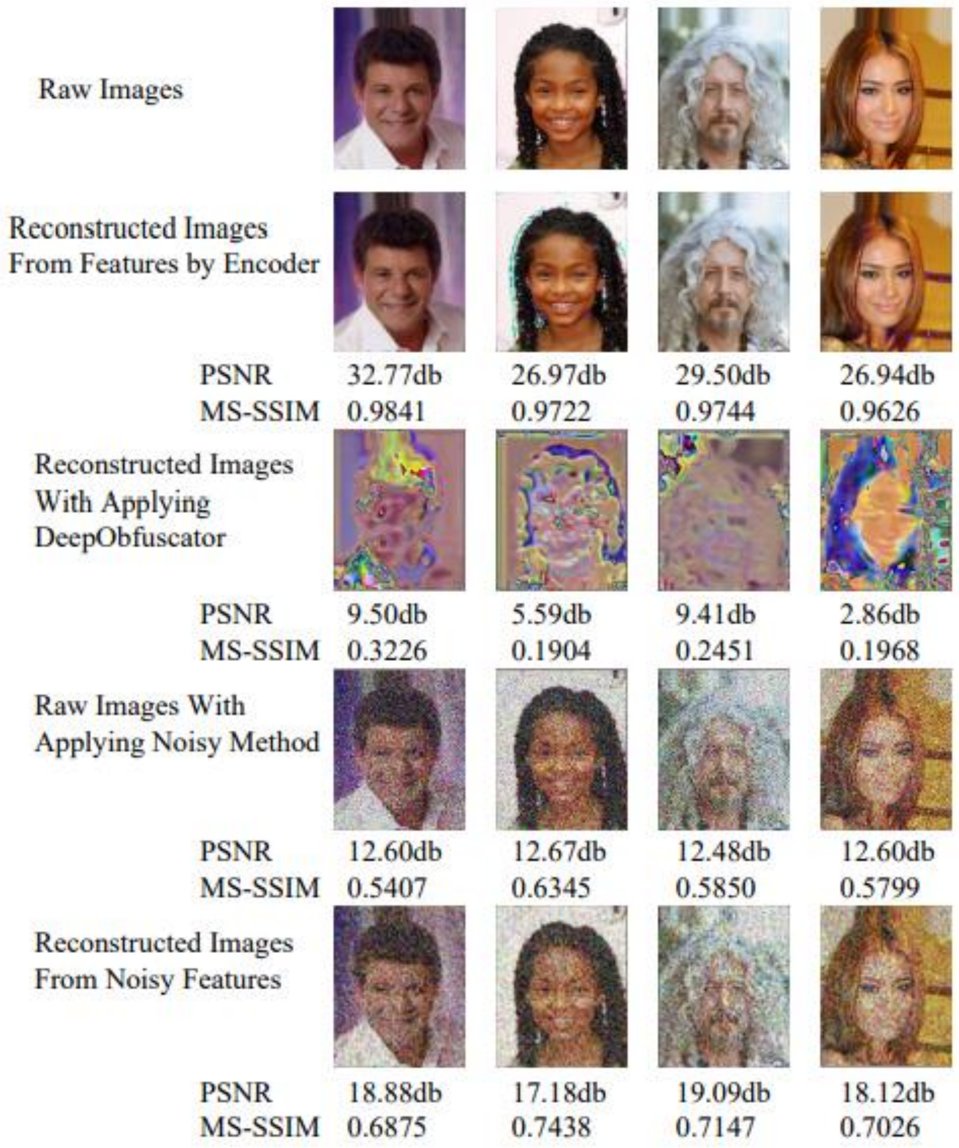
Effectiveness of Defending Against Reconstruction Attack

- Peak Signal to Noise Ratio (PSNR) is a widely used metric of image quality is used to evaluated the quality of reconstructed images

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

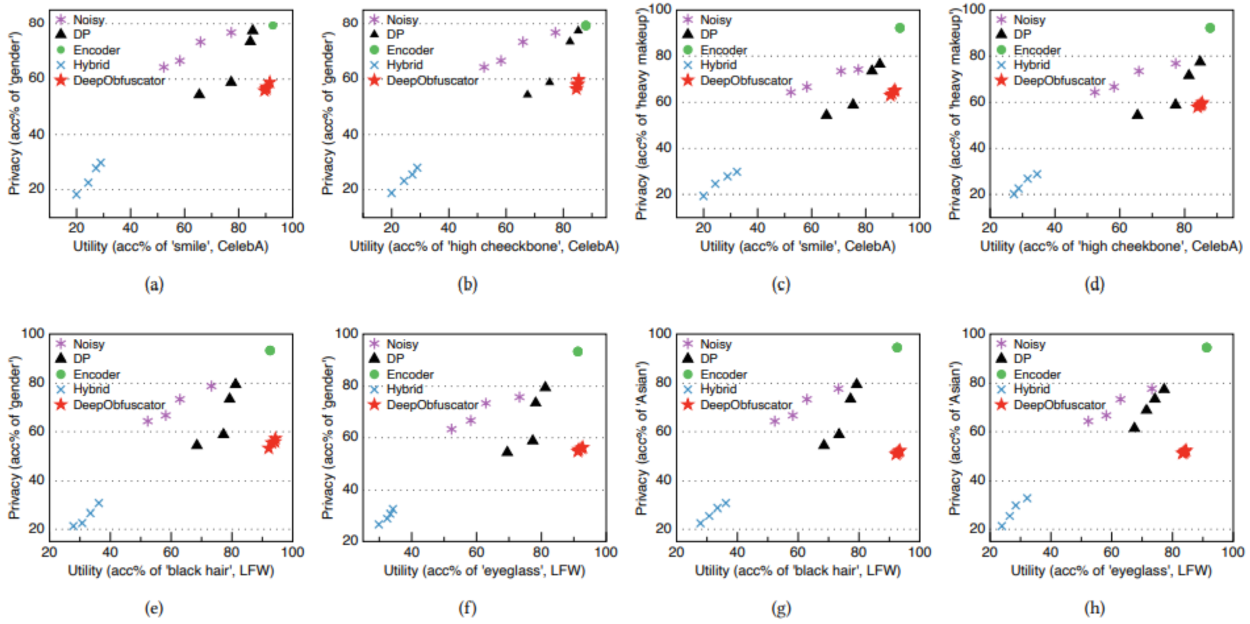
Comparison


Raw Images
Reconstructed Images From Features by Encoder
PSNR 32.77db 26.97db 29.50db 26.94db
MS-SSIM 0.9841 0.9722 0.9744 0.9626
Reconstructed Images With Applying DeepObfuscator
PSNR 9.50db 5.59db 9.41db 2.86db
MS-SSIM 0.3226 0.1904 0.2451 0.1968
Raw Images With Applying Noisy Method
PSNR 12.60db 12.67db 12.48db 12.60db
MS-SSIM 0.5407 0.6345 0.5850 0.5799
Reconstructed Images From Noisy Features
PSNR 18.88db 17.18db 19.09db 18.12db
MS-SSIM 0.6875 0.7438 0.7147 0.7026

Human Perceptual Study

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

Effectiveness of Defending Against Private Attribute Leakage



Effectiveness of Defending Against Private Attribute Leakage

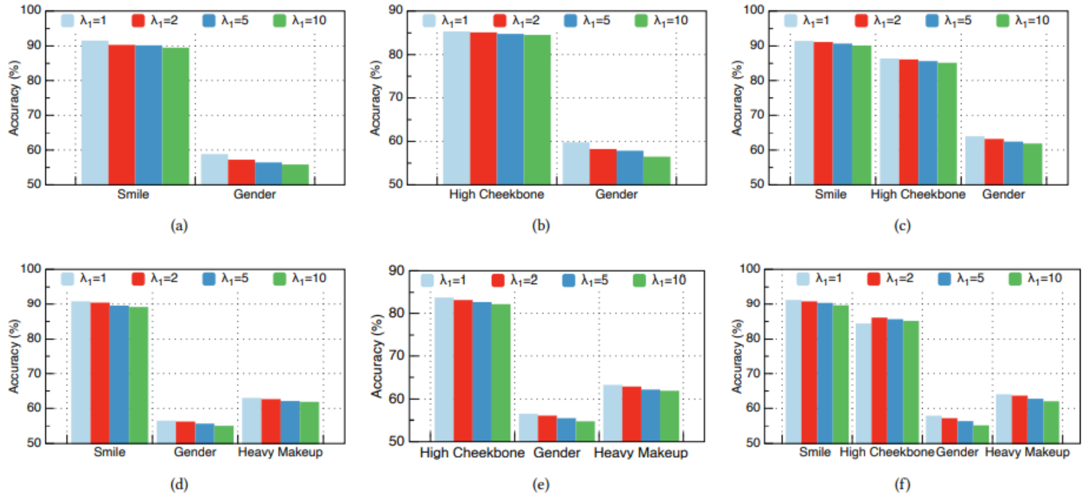


Figure 7: The impact of the utility-privacy budget λ_1 on CelebA. (λ_2 = 1)

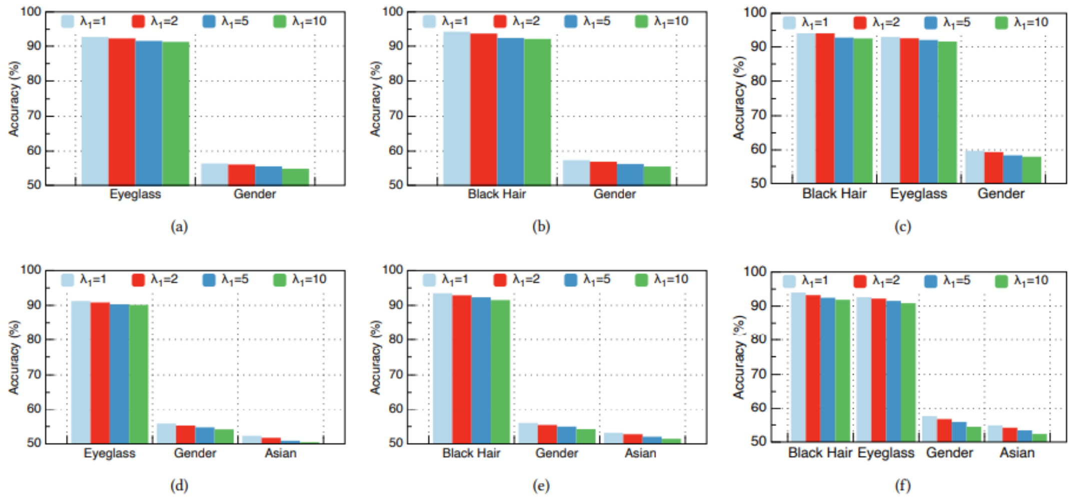


Figure 8: The impact of the utility-privacy budget λ_1 on LFW. (λ_2 = 1).

Cross-Dataset Evaluation

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%

Results

Experimented on CelebA and LFW datasets. Results: quality is dramatically decreased from 0.9458 to 0.3175 in terms of multi-scale structural similarity, so person in image is hard to identify. Classification accuracy of private attributes is reduced to a random guessing level 97.36% to 58.85%. Cloud services only reduce by 2%.

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

Performance on Smartphones

Discussion

- Did not perform the model optimization for the obfuscator in terms of efficiency.
- DeepObfuscator can be extended to other data modalities such as sensor data (accelerometer, gyroscope)
- Even though DeepObfuscator attains a notably better privacy-utility tradeoff than existing works, it requires prior knowledge of primary learning tasks before training.
    - This requirement lay limit applicability and generalization in practice