Visual Privacy Protection in Mobile Image Recognition Using Protective Perturbation

Mengmei Ye*
IBM Research
Yorktown Heights, NY, USA
mye@ibm.com

Zhongze Tang
Rutgers University
Piscataway, NJ, USA
zhongze.tang@rutgers.edu

Huy Phan
Rutgers University
Piscataway, NJ, USA
huy.phan@rutgers.edu

Yi Xie
Rutgers University
Piscataway, NJ, USA
yi.xie@rutgers.edu

Bo Yuan
Rutgers University
Piscataway, NJ, USA
bo.yuan@soe.rutgers.edu

Sheng Wei
Rutgers University
Piscataway, NJ, USA
sheng.wei@rutgers.edu

ABSTRACT

Deep neural networks (DNNs) have been widely adopted in mobile image recognition applications. Considering intellectual property and computation resources, the image recognition model is often deployed at the service provider end, which takes input images from the user's mobile device and accomplishes the recognition task. However, from the user's perspective, the input images could contain sensitive information that is subject to visual privacy concerns, and the user must protect the privacy while offloading them to the service provider. To address the visual privacy issue, we develop a protective perturbation generator at the user end, which adds perturbations to the input images to prevent privacy leakage. Meanwhile, the image recognition model still runs at the service provider end to recognize the protected images without the need of being re-trained. Our evaluations using the CIFAR-10 dataset and 8 image recognition models demonstrate effective visual privacy protection while maintaining high recognition accuracy. Also, the protective perturbation generator achieves premium timing performance suitable for real-time image recognition applications.

CCS CONCEPTS

• **Security and privacy → Systems security**; • **Information systems → Multimedia information systems**.

KEYWORDS

Visual privacy, image recognition, adversarial perturbation

ACM Reference Format:
Mengmei Ye, Zhongze Tang, Huy Phan, Yi Xie, Bo Yuan, and Sheng Wei. 2022. Visual Privacy Protection in Mobile Image Recognition Using Protective Perturbation. In *13th ACM Multimedia Systems Conference (MMSys '22), June 14–17, 2022, Athlone, Ireland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3524273.3528189

*The work was completed during the author's PhD study at Rutgers University.

MMSys '22, June 14–17, 2022, Athlone, Ireland

ACM ISBN 978-1-4503-9283-9/22/06…$15.00
https://doi.org/10.1145/3524273.3528189

1 INTRODUCTION

Deep neural network (DNN)-based image recognition is an important and popular category of mobile applications that has been adopted in various domains. For example, eBay Image Search and Google Lens [3, 4] can identify objects in the input images collected/generated by the mobile device and provide the mobile users with metadata of the detected objects (e.g., purchase links and customer reviews) in real time. In healthcare systems, image recognition has been adopted to assist medical diagnosis [21]. However, the input images in such applications often contain sensitive information (e.g., human faces, license plates, and medical records), which raises visual privacy concerns [54] when the image recognition task is handled by third-party service providers due to the considerations of intellectual property and computation resources. In addition, the sensitive information is often subject to privacy regulations (e.g., HIPAA [5] and GDPR [2]), which have strict restrictions on the presence of user data in different geographic or administrative locations after leaving the user device. Such privacy concerns or regulations have significantly impacted the deployment and utilization of third-party services for the computation-intensive DNN applications, which eventually results in downgraded application performance and user experience (e.g., if the application must be run locally with low-end computation resources) or sophisticated legal implications [10]. Therefore, it is crucial to take privacy protection into consideration and minimize the exposure of sensitive information in the image recognition applications.

1.1 The Visual Privacy Problem

In particular, the direct privacy issue with regard to image recognition is visual privacy [31, 54, 59]. In a nutshell, visual privacy is a psychological concern of the user (i.e., owner of the image) that the image containing privacy-sensitive information may be exposed to another person. Such visual privacy concern is not a new psychological effect as the same would exist with the physical presence of sensitive information in the pre-digital and pre-Internet era; however, the recent advancements of cloud computing, digital and social media, as well as deep learning applications have inevitably promoted visual privacy as a prevalent concern, especially when large volumes of sensitive images are delivered to third parties for deep learning services. Different from many other data privacy problems, visual privacy has two unique properties that cause significant challenges in terms of the privacy protection:

- *No specific privacy-sensitive features.* It is difficult to define or identify the privacy sensitive features in an image that is subject to visual privacy issues. The user is generally concerned about the entire image being exposed, which may involve all or certain features varying by the users and significantly enlarge the potential attack surface that requires visual privacy protection.
- *No specific adversaries.* As a psychological concern, there are typically no clearly defined adversaries in an event of visual privacy exposure; instead, the victim user is simply concerned about the visual privacy leakage to any other person, which significantly challenges the defense approach.

Consequently, the visual privacy problem is significantly different and more challenging than general privacy/security issues, in that the protection must cover *all* identifiable features in the image against *all* individuals other than the original owner of the image.

1.2 The State-of-the-Art Solutions

Several approaches have been developed in the community to protect the privacy of sensitive images. One category of approaches partitions the image recognition model into a sensitive component and a non-sensitive component, which are deployed at the trusted entity (e.g., the user end) and the untrusted entity (e.g., the service provider end), respectively [13, 18, 48]. However, in reality, the deep learning-based image recognition models are often proprietary assets with intensive computations, which are challenging to be deployed, even partially, at the user end.

To address this limitation, another category of approaches focuses on proactive image protection. For example, homomorphic encryption-based approaches [32, 40] encrypt the original image to eliminate information exposure without requiring the service provider to decrypt the protected image for the required computations. In this case, the privacy sensitive information is never exposed to the untrusted parties for image recognition. However, homomorphic encryption would incur high performance overhead in both the training and inference phases, which is hardly applicable to real-time image recognitions.

To balance between privacy protection and system efficiency, several recent works leverage differential privacy and image blurring/pixelation to conceal the sensitive features in the images [16, 38, 44, 51]. However, the effectiveness of these approaches relies on the accurate identification of all the sensitive features in the target image for hiding and blurring, which is technically difficult to achieve and hardly adaptable to different applications that consider distinct features as sensitive information.

1.3 Proposed Solution: Protective Perturbation

To address the aforementioned limitations in the state-of-the-art approaches, we develop a novel privacy protection approach for image recognition applications using protective perturbation. Our proposed protective perturbation is a proactive pixel mask being added to the privacy-sensitive input image, which achieves two objectives: (1) the target image is blurred to eliminate the exposure of sensitive information (i.e., *invisible* to human vision); and (2) the blurred image can still be directly adopted by the image recognition model for the original deep learning computations with no significant accuracy degradation (i.e., *visible* to machine vision). Such protective perturbations are generated by a real-time generative neural network model, which progressively optimizes for a loss function to minimize the similarity between the original and perturbed images (i.e., visibility to human vision) and maximize the accuracy of the image recognition model (i.e., visibility to machine vision). In a nutshell, our proposed protective perturbation approach aims to achieve both the goals of image blurring [16, 38, 44, 51] and computing on encrypted data (e.g., homomorphic encryption [32, 40]) but without being subject to their technical difficulties and limitations. For example, the protective perturbation does not rely on the identification of individual features (i.e., required in traditional image blurring) and thus can be made universal to adapt to different applications with distinct sensitive features. Also, the perturbed images can be directly handled by the deep learning image recognition model, as if they were the original images, without increasing the complexity and execution time of the computation.

To achieve the aforementioned design goals, our protective perturbation generator applies a generative neural network model to add perturbations to the original input images. The loss function to train the generative model contains three components, including the accuracy loss of the original image recognition model (i.e., the target model), an auxiliary model that mimics human vision and helps perturb the original image, and the similarity loss between the original image and the perturbed image. The successfully protected images should maintain high prediction accuracy on the target model (i.e., high machine vision) without retraining, low prediction accuracy on the auxiliary model, and low similarity comparing with the original images (i.e., low human vision). In our empirical evaluations on 8 target models with 7 auxiliary models and the SSIM metric (i.e., structural similarity index measure) [52], the proposed method can generate the protective perturbations successfully in real-time . To summarize, we have made the following key technical contributions in the paper:

- We propose a novel protective perturbation-based approach to achieve privacy preserving image recognition, which outperforms the state-of-the-art approaches by its universality to distinct sensitive features and low complexity;
- We implement the perturbation generator using a generative neural network model and deploy it in real systems; and
- We conduct comprehensive evaluations on the implemented perturbation generator using 8 popular image recognition models to justify its effectiveness and real-time performance.

2 BACKGROUND

2.1 DNN-based Image Recognition System

Figure 1 illustrates an image recognition system targeted by this paper, which identifies the class of an input image by leveraging deep learning. First, the user provides the input image and sends it to the service provider. Then, the service provider leverages a deep learning-based image recognition model to recognize the image. The model contains an input layer, hidden layers, and an output layer, which is trained by a training dataset and a validation dataset with parameter tuning. The input layer processes the input image and sends the intermediate results to the neurons (i.e., nodes in Figure 1) with weights (i.e., edges in Figure 1) in the hidden layers.

After that, the output layer outputs the recognition confidence for each class (e.g., 1% as a dog, and 91% as a car). Finally, the service provider sends the image recognition result (i.e., the class with the highest recognition confidence) to the user.

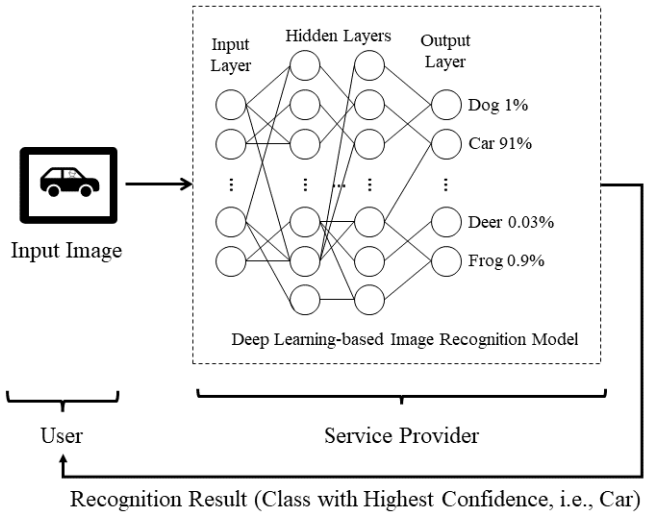


Figure 1: Deep learning-based image recognition system.

2.2 Privacy Issues in Image Recognition

In such image recognition system, there exist privacy concerns with regard to the input image that may contain sensitive information [16, 38, 44, 51]. When the user sends the input images to the untrustworthy service provider, the service provider may gain access to the user's private information even under strong security protections. For example, even if the end-to-end system is protected by the state-of-the-art system and network security mechanisms (e.g., Trusted Execution Environment [1, 6] and end-to-end encryption [17]), the protected input images would still need to stay in the clear right before or during the execution of the image recognition model and thus exposed to the service provider. Also, even if the sensitive information, such as human face and license plate number, is blurred for privacy protection, the image background and user's outfit could still potentially disclose privacy-sensitive locality or demographic information.

In this case, even if the service provider does not have any malicious intent, the exposure of the user private information still constitutes a breach of user privacy. In other words, the special distinction between security and privacy requirements leads to incompetence of the state-of-the-art security techniques to protect user's privacy, which creates a unique privacy problem that must be addressed separately from all the existing security measures. In summary, we consider that various types of information in the input images fed into the image recognition system could cause privacy concerns when exposed to an untrusted party. It is crucial to clearly identify and define these potential threats to the user privacy and proactively address them to achieve a privacy preserving image recognition system.

2.3 Security and Privacy Threat Models

To summarize the discussions about the security and privacy of image recognition systems, we define the following security and privacy threat models as the target of this paper.

- **Security threat model**. We assume that the state-of-the-art security measures have been deployed to secure the end-to-end image recognition system, which includes but not limited to trusted execution environments (e.g., TrustZone [1] and SGX [6]) on both the user-end and provider-end systems, as well as end-to-end data encryption [17]. In addition, we assume that the untrusted party (i.e., the service provider) does not have the incentive or capability to breach such strong security measures or employ other advanced security attacks (e.g., side channel attack or reverse engineering attack) to proactively infer the user's private information. Such security related assumptions are attributed to the ever increasing security awareness of the service providers and the deployment of state-of-the-art system and network security measures to protect the image recognition service, most of which have been constantly discussed in the security community.
- **Privacy threat model**. In spite of the strong security guarantee (and thus weak security threats), we consider a strong privacy threat model in this paper, which cannot be eliminated by the state-of-the-art security protection mechanisms. In particular, we define a privacy breach as the exposure of users' private images to another party other than the users themselves, which are visible and perceivable by human eyes. In this case, the user's privacy concern originates from the unknown and unpredictable consequences of private information being exposed to others, even without malicious intent or actual damages involved. Following this privacy threat model, the service provider in the image recognition system, even without malicious intent and with the strongest security protection deployed, is considered as an "adversary" for potential privacy breaches, simply because the image recognition tasks eventually require the service provider's access to the original input images in the clear.

Note that this paper assumes that the user-end system is both secure and trustworthy (i.e., with no security and privacy concerns), and we only consider preventing privacy leakage from the input images when they leave the user device. Also, other security and privacy threat models targeting information leakage at other phases, such as when training the deep learning models or after retrieving the image recognition results, are actively discussed in the deep learning community and thus out of the scope for this paper.

3 PROTECTIVE PERTURBATION

3.1 System Overview

To address the privacy issues in image recognition applications, we develop a protective perturbation generator at the user end to proactively add human-visible but machine-invisible masks to the input images. Such protective perturbations would help hide the privacy-sensitive information from being exposed to adversaries but still maintain the original functionality of the image recognition application. Figure 2 shows the overall workflow of the proposed

privacy-preserving image recognition system involving three parties with various system components and operations: (1) at the *user* end, the proposed *perturbation generator* processes the original input images and injects the protective perturbations before offloading it to the service provider for image recognition; (2) at the *service provider* end, since the injected protective perturbation is transparent to the target deep learning-based image recognition model, the image recognition computations would proceed as normal without impacting the accuracy and speed of the inference; and (3) the *adversary* attempts to gain access and breach the privacy-sensitive information in the perturbed images; however, such efforts are expected to fail given the protective perturbations.

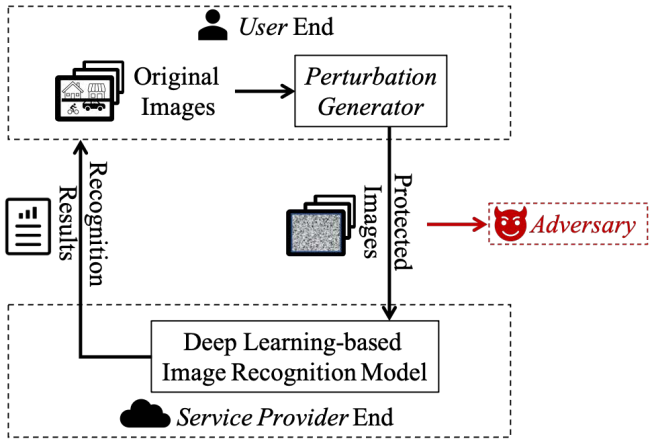


Figure 2: The system workflow of privacy-preserving image recognition with the proposed perturbation generator.

In the end-to-end workflow, the *perturbation generator* plays a central role in adding privacy-preserving protective perturbations, which are expected to be *non-intrusive* to both the *user* and the *service provider* of the original image recognition application, but *intrusive* to the adversaries:

- The *non-intrusive* objective for the *user* and the *service provider* are two-fold: (1) from the perspective of *model deployment*, the protective perturbation-based approach should eliminate the need to re-train or alter the original image recognition model in any form, presenting an *non-intrusive* privacy-preserving solution that can be immediately deployed in the commodity systems today; and (2) from the perspective of *runtime execution*, the protective perturbation should maintain the original accuracy of the image recognition and, more importantly, the additional delay incurred by the perturbation generation should not be noticeable by the *user* or jeopardise the real-time requirement of the original application.
- The *intrusive* objective for the *adversary* requires that the input images obfuscated by the protective perturbations are not perceivable to human vision, defeating the adversary's intent to obtain any privacy-sensitive information.

The above *non-intrusive* and *intrusive* objectives appear to be opposite and contradicting to each other. However, there exists a clear distinction between the contexts of the two objectives in that the *non-intrusive* objective targets machine learning models/algorithms, while the *intrusive* objective targets human vision. Even though the goal of machine vision has always been to approach and emulate human vision, there is still a big enough gap between them, which can be leveraged to achieve distinct effects on the two vision systems with the same set of inputs. As an evidence, the recent developments in the AI community on adversarial attacks [12, 27, 33, 36] could successfully generate perturbations to the input data that lead to erroneous inference results by the machine but remain unnoticeable by human. This clearly justifies the feasibility of leveraging the gap between machine and human visions to achieve the distinctive objectives and thus forms the foundation of our proposed protective perturbation generation method, as described next.

3.2 Protective Perturbation Generation

Figure 3 shows our proposed protective perturbation generation process. We first preprocess the original image x_i (e.g., by random cropping, random horizontal flipping, and normalization). Then, we feed x_i to a generative neural network model, namely U-Net [39], which has been adopted for generating traditional adversarial perturbations [36, 37]. We define the U-Net model as $U(\cdot)$, and it generates the perturbation δ that aims to hide image features. With $clamp(\cdot)$ that clips $x_i + \delta$ into a valid pixel range, we have the protected image $x_i' = clamp(x_i + \delta)$ that protects user privacy. Using only one forward propagation, U-Net avoids time-consuming iterations to produce outputs, which makes real-time generation possible.

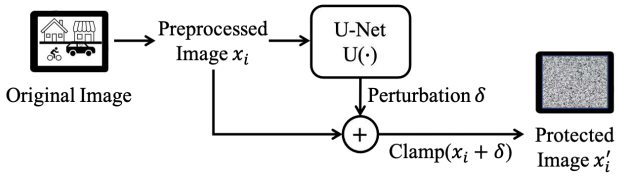


Figure 3: The process of generating a protected image.

The major challenge in applying U-Net to our *protective* perturbation generation lies in the aforementioned *intrusive* and *non-intrusive* objectives that are opposite to the traditional *adversarial* attacks. In particular, our protective perturbation aims to *maximize* the perturbation to human vision and *minimize* the impact to the image recognition model, while the U-Net model for traditional adversarial attack aims to *minimize* the former and *maximize* the latter. To summarize, we have two objectives in the design of the protective perturbation generator, which are consistent with the discussions in Section 3.1:

- *Non-intrusive* objective - *Maintain the machine vision*, which can be formulated as
  $Accuracy(F_{target}(x_i')) \rightarrow Accuracy(F_{target}(x_i))$,
  assuming the original image recognition model is the target model $F_{target}(\cdot)$; and
- *Intrusive* objective - *Maximize the human-perceivable perturbation*, which can be formulated as $Accuracy(F_{adv}(x_i')) \rightarrow 0$, assuming $F_{adv}(\cdot)$ mimics the adversarial recognition.

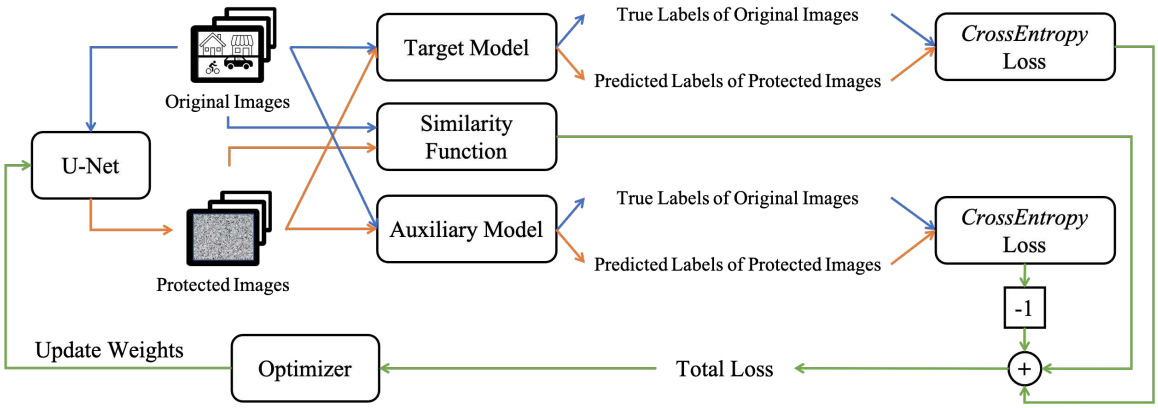


Figure 4: The training process of the perturbation generator. After it reaches *max_epochs*, we select the best trained U-Net model as the protective perturbation generator.

The following subsection elaborates on how to achieve these two goals simultaneously and train an effective and efficient protective perturbation generation model $U(\cdot)$.

Algorithm 1: Training Algorithm for the Protective Perturbation Generator

Input: Training dataset $X = \{x_1, ..., x_m\}$, true labels $Y = \{y_1, ..., y_m\}$, target model $F_{target}(\cdot)$ and its loss weight ω_{target}, auxiliary model $F_{aux}(\cdot)$ and its loss weight ω_{aux}, weight of the similarity loss ω_{SI}, and the maximal epochs *max_epochs*.

Result: Trained protective perturbation generator $U(\cdot)$.

Randomly initialize $U(\cdot)$.

for $j \leftarrow 1$ **to** *max_epochs* **do**

  foreach x_i, y_i **in** X, Y **do**

    $\delta \leftarrow U(x_i)$;

    $x_i' \leftarrow clamp(x_i + \delta)$;

    $y_{pred\_target} \leftarrow F_{target}(x_i')$;

    $y_{pred\_aux} \leftarrow F_{aux}(x_i')$;

    $Loss \leftarrow$ $\omega_{target} \cdot CrossEntropy(y_{pred\_target}, y_i) - \omega_{aux} \cdot CrossEntropy(y_{pred\_aux}, y_i) + \omega_{SI} \cdot SSIM(x_i, x_i')$;

    Update $U(\cdot)$ to minimize *Loss*;

  end

end

3.3 Training the Perturbation Generator

Figure 4 shows the training process of our proposed protective perturbation generator. The blue arrows show the data flow of original images, the orange arrows indicate the data flow of protected images, and the green arrows reveal the loss and weights updates during the training process.

To be more specific, we first provide the original training images to an initial $U(\cdot)$ to generate the initial protected images. Then, we feed the original images and the protected images into the target image recognition model. The target model provides the true labels of the original images and the predicted labels of the protected images to the *CrossEntropy* loss function, which helps achieve the aforementioned *non-intrusive* objective. Let a true label be y_i and the corresponding label predicted by the target model be y_{pred_target}. We have the loss calculation as follows:

$$Loss_{target} = CrossEntropy(y_{pred\_target}, y_i) \quad (1)$$

Furthermore, we leverage two modules to achieve the aforementioned *intrusive* goal. First, we apply the original images and the protected images to an *auxiliary model*, which is an image classification model to evaluate the classification results seen by the adversary. The *auxiliary model* provides the true labels of the original images and the predicted labels of the protected images. The *CrossEntropy* loss function calculates the loss value for the auxiliary model, which is applied a negative sign to maximize the loss between the two sets of the labels and achieve the aforementioned *intrusive* goal. The loss function represented by the auxiliary model is defined as:

$$Loss_{aux} = -CrossEntropy(y_{pred\_aux}, y_i) \quad (2)$$

Then, we apply a similarity function to determine the human-perceivable difference between the original images and the protected images to ensure that the two images are visually different and prevent privacy exposure. We use the structural similarity index measure (SSIM) metric as our similarity function, since it is one of the most popular metrics to evaluate the perceivable similarity between two images for human visual perception [52]. The loss function represented by the SSIM metric is defined as:

$$Loss_{SI} = SSIM(x_i, x_i') \quad (3)$$

Finally, the optimizer obtains the total loss value and updates the weights on the U-Net model. The final loss function is formulated as follows:

$$Loss = \omega_{target} \times Loss_{target} + \omega_{aux} \times Loss_{aux} + \omega_{SI} \times Loss_{SI} \quad (4)$$

where ω_{target}, ω_{aux}, and ω_{SI} are the pre-set coefficients to control the effect of each loss component.

Algorithm 1 shows the detailed pseudocode of the presented model training process. The training keeps executing and updating

$U(\cdot)$ until it reaches *max_epochs*. Finally, a successfully protected image x_i' generated by $U(\cdot)$ aims to fulfill the following requirements at the same time:

(1) $F_{target}(x_i') = y_i$, where the predicted label of the target model is the same as the true label y_i;
(2) $F_{aux}(x_i') \neq y_i$, where the predicted label of the auxiliary model is different from the true label; and
(3) Minimal $Loss_{SI}$, where the similarity level between x_i and x_i' should be as low as possible.

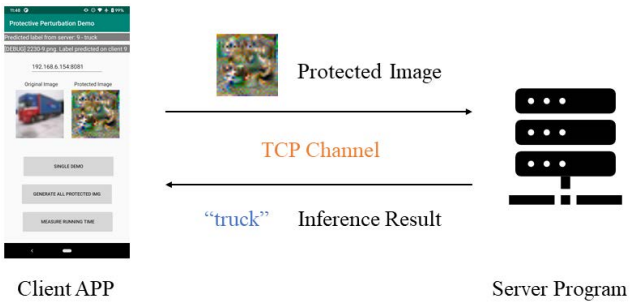


Figure 5: Mobile system implementation.

3.4 Mobile System Implementation

Figure 5 illustrates the implementation of our protective perturbation system, which consists of a mobile app running on an Android smartphone and a server running on a workstation connected via a TCP communication channel. The app integrates the perturbation generator with the image recognition application, and it executes the perturbation generator (described in Section 3.2) to perturb the input image before offloading it to the server. After receiving the protected input image, the server executes the target DNN model for image recognition and returns the inference results to the smartphone. We adopt PyTorch Mobile [8] and PyTorch [34] in the implementation of the mobile app and the server program, respectively.

3.5 Case Study

Figure 6 shows a case study of our proposed protective perturbation approach. The user intends to send a set of privacy-sensitive images (in Figure 6a) to the service provider for image recognition using ResNet34. Without applying protective perturbation, the sensitive images are all exposed to the service provider while completing the image recognition task, which is a direct breach of visual privacy as discussed in Section 2.3.

To eliminate the visual privacy issue, the user employs our proposed protective perturbation mechanism to protect the input images. In this case, the user chooses ResNet18 as the auxiliary model and trains the perturbation generator using the procedure presented in Section 3.3. Then, before offloading the images to the service provider end, the user executes the perturbation generator to add perturbations in the sensitive images (Figure 6b). As a result, the same image recognition results are returned from the service provider without any side effects (i.e., the *non-intrusive* objective is achieved), but now the images are completely invisible to human

(a) Original Images **(b) Perturbed Images**

Figure 6: Example of original images (left) and the perturbed version (right). In this case, the target model is ResNet34, and the auxiliary model is ResNet18.

eyes, which eliminates the visual privacy concern. Note that the protective perturbations blurred the entire images that may be exposed to any adversary, without being specific to certain features or adversaries, which is compliant with the unique features of the visual privacy problem discussed in Section 1.1.

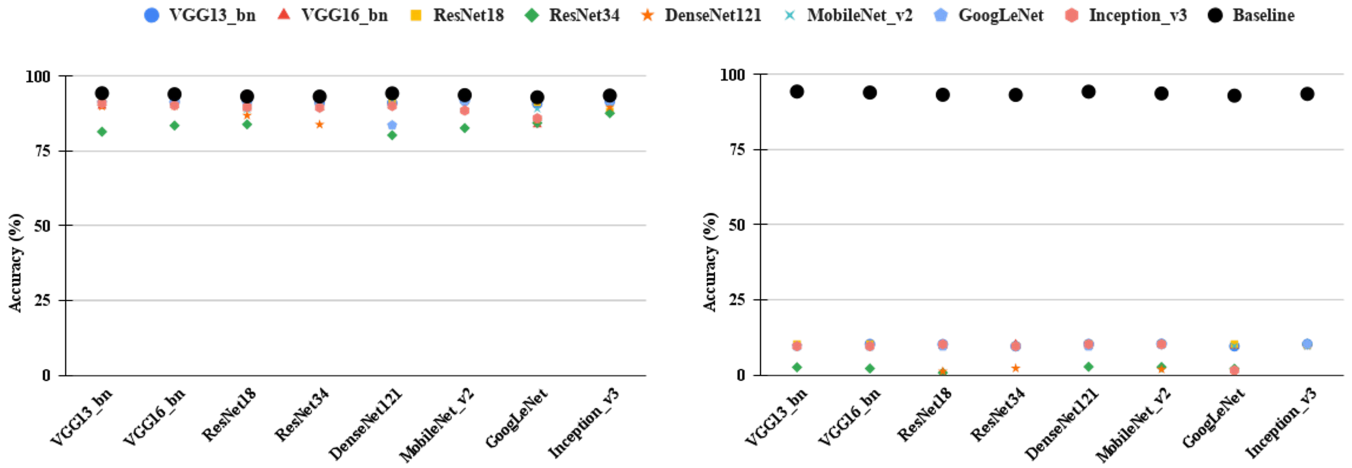
4 EXPERIMENTAL RESULTS

4.1 Implementation and Experimental Setup

4.1.1 Dataset. We train and test our protective perturbation generator based on the CIFAR-10 dataset [23], which contains 60K 32×32 RGB images in 10 classes, with 6K images per class. We use the original CIFAR-10 training dataset (50K images) and split the CIFAR-10 test dataset (10K images) into two halves as our validation dataset and test dataset. The dataset is provided by the torchvision library [28]. We preprocess the images in the dataset following [35].

4.1.2 Target and Auxiliary Models. We evaluate 8 neural network models pre-trained by [35] as our target image recognition models, which include VGG13_bn, VGG16_bn [43], ResNet18, ResNet34 [19], DenseNet121 [20], MobileNet_v2 [42], GoogLeNet [45], and Inception_v3 [46]. The properties (i.e., name, accuracy on validation dataset, number of parameters, and the size) of the 8 neural networks are listed in Table 1. For each target model, we apply the 7 other models as the auxiliary models and, therefore, we have 56 pairwise combinations of target model and auxiliary model for the experiments. The target models are those with privacy-sensitive inputs under protection, and they provide true labels of both the original inputs and the protected inputs. The auxiliary models provide another way to provide true labels of the original inputs but wrong labels of the protected inputs.

4.1.3 Training and Validation Processes. First, we define loss parameters based on Equation (4) in Section 3.3. In the loss function, ω_{target}, ω_{aux}, and ω_{SI} are set as 1.0, 1.0, and 0.5, respectively, which come from the experiments. In the training process, we use AdamW [26] as our optimizer, in which we set 0.001 as the learning rate for the GoogLeNet (target model) and VGG13_bn (auxiliary model) pair, and 0.005 for the rest of the target-auxiliary model pairs; the weight decay is 0.0005 with other parameters remaining as default. We define the batch size as 256, the maximum epoch number as 100, the number of workers as 8, and the precision bit



(a) Prediction Accuracy of Target Models

(b) Prediction Accuracy of Auxiliary Models

Figure 7: Prediction accuracy results of the target and auxiliary models on the protected images. The x-axis indicates the target models; the y-axis indicates the accuracy values (%); and the legend indicates the auxiliary models and the baseline (i.e., on the original images).

Table 1: Properties of target and auxiliary models [35]. From left to right, the four columns are: model name, the accuracy on the validation dataset, the number of parameters, and the size of the pre-trained model.

Model	Val. Acc.	No. Params	Size (MB)
VGG13_bn	94.22%	28.334 M	109
VGG16_bn	94.00%	33.647 M	129
ResNet18	93.07%	11.174 M	43
ResNet34	93.34%	21.282 M	82
DenseNet121	94.06%	6.956 M	28
MobileNet_v2	93.91%	2.237 M	9
GoogLeNet	92.85%	5.491 M	22
Inception_v3	93.74%	21.640 M	83

as 16. After the training process, the validation process selects the best trained generator with the lowest loss value among all the 100 epochs. Pytorch [34] and Pytorch Lightning [15] are adopted to train and test our proposed method.

4.1.4 Hardware Setup. In the training and validation processes, we use a *GPU workstation* with NVIDIA RTX A6000 48GB GPU, 16-core 3.0 GHz Intel i9-10980XE CPU, and 128GB RAM. We deploy the trained generator model on a Pixel 3 *smartphone* with Snapdragon 845 SoC for the effectiveness and timing evaluations. In addition, the timing evaluations of the target models and the trained generators are performed on the same *GPU workstation*, under CPU-only and GPU-accelerated settings.

4.2 Effectiveness Evaluation

We conduct the effectiveness evaluation for all the 56 experiments with the test dataset. In each experiment, we evaluate the prediction accuracy for both the target model and the auxiliary model on the protected images generated by the trained protective perturbation generator.

4.2.1 Evaluation Metrics. We apply the following two metrics to evaluate the effectiveness of the protective perturbations.

- *Prediction accuracy*: As shown in Equation (5), the accuracy calculates a matching rate of true labels and predicted labels for the protected images. The baseline results indicate the accuracy for the original images. The implementation of the accuracy calculation is based on [7].
- *SSIM* [52]: SSIM is a widely adopted metric for the quality of images. In this case, a lower SSIM value indicates less similarity between an protected image and an original image and, therefore, the lower the SSIM value is, the more difficult for human to recognize (i.e., better privacy protection). Considering that a negative SSIM value means an inverted structure of the image [53], the SSIM values we present in the evaluations are absolute values. The implementation of SSIM is based on [9].

$$Accuracy = \sum_{\substack{0<i\leq N \\ Pred_i=Label_i}} \frac{1}{N} \times 100\% \quad (5)$$

4.2.2 Evaluation Results. Figure 7(a) and 7(b) present the prediction accuracy of the target models and the auxiliary models, respectively. There are 56 pairwise experiments with the 8 target models and 7 auxiliary models each. Among them, there are 38 experiments achieving high accuracy of the target models that only reduces less than 5% compared to the baseline case, justifying the *non-intrusive* objective for machine vision. For the accuracy of the auxiliary models, the results are between 0.74% and 10.34%, which

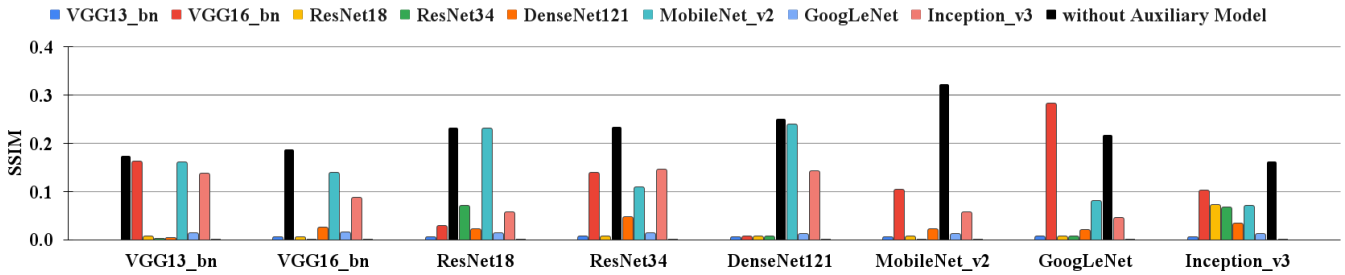


Figure 8: SSIM results on the protected images. The x-axis indicates the target models; the y-axis indicates the SSIM values; and the legend indicates the auxiliary models and the case without auxiliary model (i.e., black bars).

Table 2: Target model accuracy of the protective perturbation generator with and without auxiliary models.

Target Model	Auxiliary Model			No Auxiliary
	Min	Median	Max	
VGG13_bn	81.36%	90.86%	92.08%	92.58%
VGG16_bn	83.42%	91.34%	91.90%	91.48%
ResNet18	83.82%	89.18%	91.66%	91.22%
ResNet34	83.76%	90.98%	91.54%	91.82%
DenseNet121	80.20%	90.26%	91.74%	92.08%
MobileNet_v2	82.58%	88.88%	91.80%	91.84%
GoogLeNet	83.74%	85.84%	91.22%	90.08%
Inception_v3	87.56%	89.62%	91.62%	91.28%

indicate that the auxiliary models fail to recognize more than 89% of the protected images, justifying the *intrusive* objective for human vision.

Furthermore, to evaluate the role of the auxiliary model in perturbation generation, we compare the accuracy of the target models with and without auxiliary models to train the generators, as shown in Table 2. In the "with auxiliary model" case, most of the target model accuracy results are slightly lower than the "no auxiliary model" case, as the auxiliary models are intended to strengthen the perturbations. However, the corresponding SSIM values with auxiliary models are significantly lower than without auxiliary models in most target-auxiliary model pairs, as shown in Figure 8, which indicates the effectiveness of the auxiliary model in the loss function, i.e., to achieve the *intrusive* objective for human vision.

To further illustrate the significance of the auxiliary models, we present the protected image samples in Figure 9, where (a)-(h) show the images from the generators without (left) and with (right) the auxiliary models. The auxiliary models at the right subset of images are selected by the lowest SSIM values. We observe that the right subset of images in Figure 9(a)-(h) contains significantly more successful protective perturbations than the left subset. Specifically, most of the right subset meets the requirements of high accuracy on the target models, low accuracy on the auxiliary models, and low SSIM values.

In addition, we compare the effectiveness of different auxiliary models. Figure 10 presents the average target model accuracy results for each auxiliary model. We observe that VGG13_bn is the best auxiliary model to maintain the high accuracy for the target models, while ResNet34 compromises the target model accuracy more than other auxiliary models. Figure 11 presents the average SSIM results for each auxiliary model, which indicate that VGG13_bn is the best auxiliary model to minimize the similarity between the original images and the protected images. Also, MobileNet_v2 performs the worst on achieving low SSIM values for the protected images.

Table 3: Timing evaluations (ms) of the target models and their corresponding protective perturbation generators.

Target Model	CPU Time (ms)		GPU Time (ms)		Mobile Time (ms)
	DNN	Perturb. Gen.	DNN	Perturb. Gen.	Perturb. Gen.
VGG13_bn	1.116	1.355	0.012	0.047	10.717
VGG16_bn	1.262	1.324	0.012	0.047	10.840
ResNet18	0.540	1.349	0.018	0.047	10.706
ResNet34	0.827	1.392	0.027	0.047	10.824
DenseNet121	2.769	1.385	0.077	0.047	10.519
MobileNet_v2	3.580	1.210	0.033	0.048	10.735
GoogLeNet	10.636	1.195	0.045	0.048	10.747
Inception_v3	16.288	1.168	0.064	0.047	10.801

4.3 Overhead Evaluation

4.3.1 Generator Size. Our protective perturbation generator involves 7.766 M parameters, and the parameter size is only 31 MB, which can be considered as a lightweight model compared to most of the target models presented in Table 1. The generator is based on U-Net, and it inherits the advantage as a fully convolutional network to accept different sizes of inputs. In other words, when our protective perturbation is applied to another dataset with larger-size input images (e.g., 224×224 in ImageNet[14]), the number of parameters and the model size would not grow with the image size, maintaining a scalable perturbation generator.

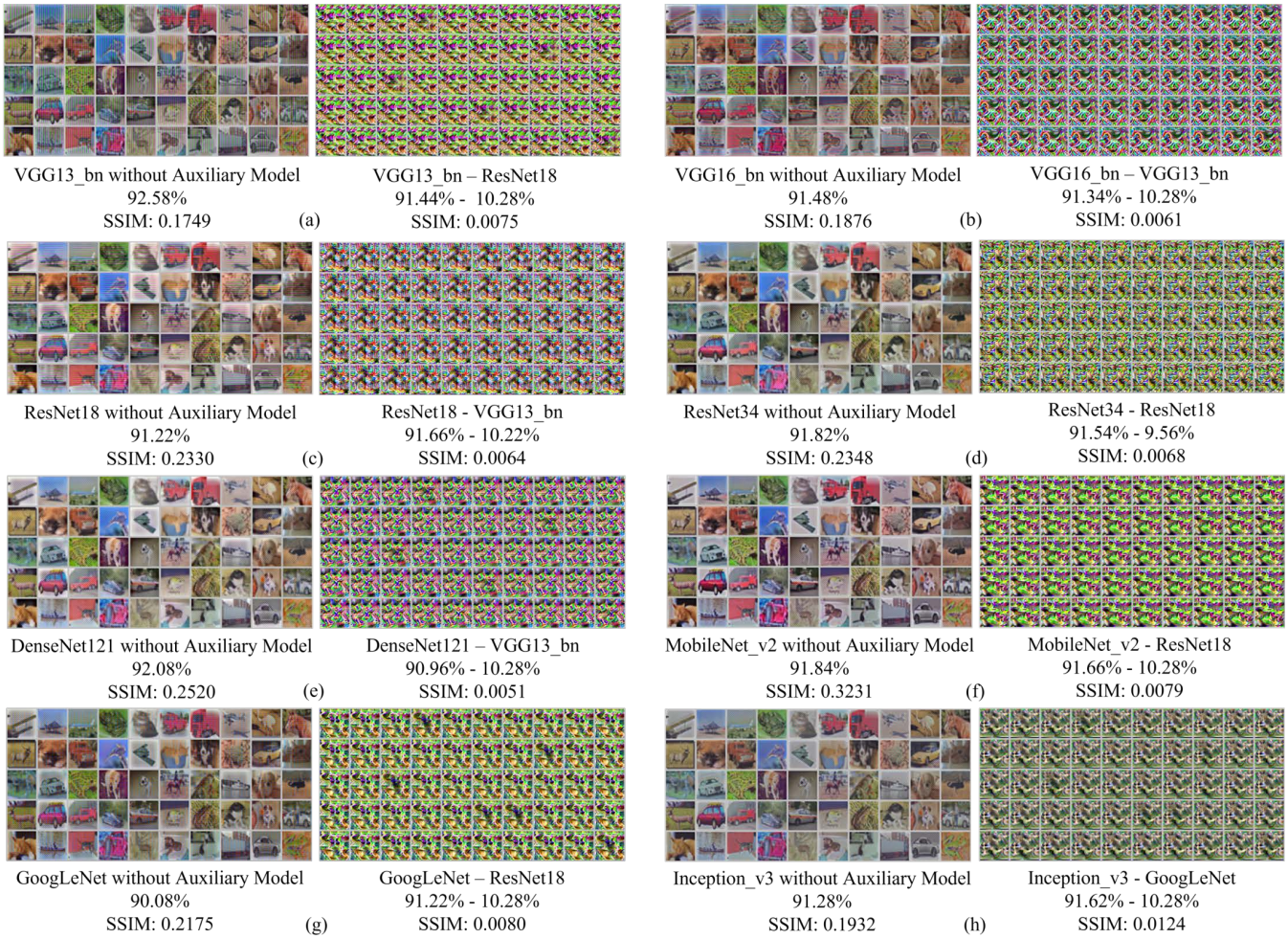


Figure 9: Samples of protected images generated from the test dataset. Each subfigure shows the results for one target model in two cases of the perturbation generator: (1) without auxiliary model (left group); and (2) with a specific auxiliary model (right group). In the description of each image group, the first line introduces the target and auxiliary model information; the second line presents the accuracy results of the target model and the auxiliary model (if the auxiliary model exists) for the test dataset; and the third line shows the SSIM values for the test dataset.

4.3.2 Timing Overhead. We conduct a comprehensive set of timing evaluations for the efficiency of the proposed perturbation generation approach. The timing evaluations are carried out under a CPU-only case, a GPU-accelerated case, and on a smartphone (described in Section 4.1.4) for comparison. We collect the execution times of the protective perturbation generators for 8 target models. On the workstation, for each target model, we calculate the average running time of the 8 generators with different auxiliary models. On the mobile phone, we measure the running time of one generator for each target model, with DenseNet121 as the fixed auxiliary model. The corresponding results are shown in the Columns 3, 5 and 6 in Table 3. In the CPU-only case, it takes 1.168 *ms* to 1.392 *ms* to generate a protected image, while in the GPU-accelerated case, it only takes 0.047 *ms* to 0.048 *ms*. On the smartphone, the generation of a protected image takes 10.519 *ms* to 10.840 *ms*. These results indicate that our proposed protective perturbation approach has a potential to be leveraged in real-time video streaming scenarios as well, in addition to individual image recognition. In addition, we also evaluate the performance of all the target models for comparison, as shown in Columns 2 and 4 in Table 3. The generator runs faster than half of the target DNN models under test (i.e., DenseNet121, MobileNet_v2, GoogLeNet and Inception_v3) in the CPU-only case. In the GPU-accelerated case, the generator is faster than DenseNet121 and Inception_v3 and has almost the same timing performance as GoogLeNet. More importantly, the results indicate that the execution time of the generator does not increase with the the growing complexity and scale of the target model, which demonstrates the advantage and feasibility of deploying the proposed approach on the user's end. Overall, the timing evaluations reveal the following:

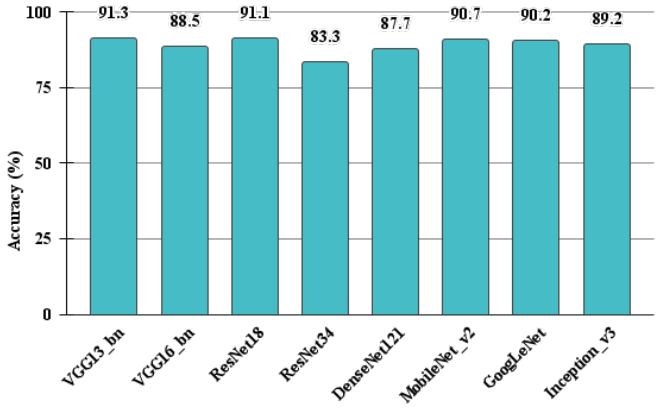


Figure 10: Average target model accuracy on the protected images with auxiliary models.

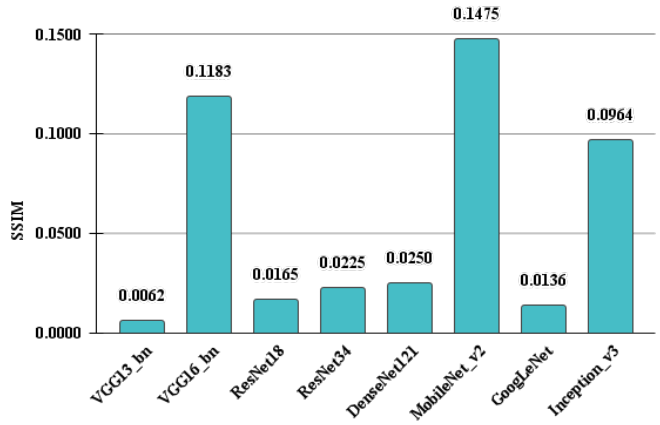


Figure 11: Average SSIM values on the protected images with auxiliary models.

- The protected images can be generated in real-time on CPU, GPU and even mobile devices, which indicates that, besides image classification applications, it is also feasible to deploy our proposed method in privacy-preserving real-time video streaming.
- The execution time of the perturbation generator is not dependent on the complexity of target models, indicating the potential of larger-scale deployment for real-world DNN models.

5 DEPLOYMENT CONSIDERATIONS

5.1 Application Scenario

Privacy protection is crucial in many real-world systems and application scenarios, with the obvious endangerment of data leakage and the corresponding law enforcement efforts [2, 5]. As long as people are exchanging information with other entities, the risk of unattended disclosure of sensitive data would exist. In this paper, we focus on solving such critical privacy concerns in the situation where the user must send privacy sensitive images to an untrusted service provider for recognition or search [3, 4, 21]. A protective perturbation generator can be deployed on the user end to add the perturbations to the images. A perturbed (and thus protected) image is blurred completely so that the adversary cannot extract any private information from it (i.e., invisible to human vision). Moreover, the protected image can still be fed into the original image recognition model (i.e., the target model) without breaking its original functionality (i.e., visible to machine vision). Such client-side deployment is non-intrusive to the already deployed DNN models, which does not require remodeling or termination of services at the provider end.

5.2 Deployment Modes

While the perturbation generator is deployed on the user end, it can be trained by either the service provider or the user, leading to two deployment modes depending on whether the target model is accessible to the user and the resource limitations for training at the user end.

- **Deployment Mode 1: Proprietary target model.** If the target model is a proprietary intellectual property of the the service provider, it would not be accessible by the end user to train the perturbation generator. However, in this scenario, it is the best interest of the service provider to train the perturbation generator and distribute it to the user as part of the product or service. It is because privacy protection has already become an integral part of many products and services that interact with users' private data, and proactively providing such protection would lead to wider deployment of the target service compliant with the privacy regulations.

- **Deployment Mode 2: Open-source target model.** If the target model is open source and fully accessible to the user, either the user or the service provider can train the perturbation generator. The user end training can be carried out offline before the adoption and execution of the third-party DNN model, which does not impact the user-end runtime performance and, in the meantime, would provide the users with higher confidence about the privacy protection of their data.

6 RELATED WORK

The proposed protective perturbation approach is related to and benefits from two domains of knowledge and developments, including privacy preserving image recognition (from the application perspective) and adversarial attacks (from the perspective of the key technical approach).

6.1 Privacy Preserving Image Recognition

Several solutions have been proposed in the community to protect the privacy in image recognition, including three categories:

Model Partitioning. Several research works [13, 18, 48] have shown that the input image and the intermediate results during the DNN execution could contain user's privacy-sensitive information. To maintain the private inference, partial layers that could leak user privacy can be deployed at the user end. However, the model could be trained by a confidential dataset, which is considered as a proprietary asset [11]. Under this scenario, all the layers in the

model are required to execute at the service provider end. Therefore, further privacy-preserving solutions are required to resolve the challenge of protecting both the user privacy and the model intellectual property.

Homomorphic Encryption. Another privacy protection approach is homomorphic encryption, which is different from the traditional encryption in that it does not require decryption to perform the computations. By leveraging homomorphic encryption, the service provider can perform the image recognition tasks (e.g., face recognition) on the encrypted images directly, and only the user can access the decrypted images [32, 40]. However, these methods result in high computational overhead, which impacts the efficiency of the image recognition system.

Image Blurring/Pixelation. Image blurring/pixelation has been one of the popular solutions to protect the sensitive features in the image. For example, researchers have applied differential privacy to pixelate the sensitive features to protect user privacy [16, 51]. Also, the works presented in [38, 44, 54, 58] blur the faces, license plates or other sensitive features in the images/videos to avoid privacy exposure. However, these methods only protect specific sensitive features in the image and cannot be applied for protecting the images containing a variety of arbitrary sensitive features in the user's mind. Furthermore, noise-based training approaches have been proposed [50], which adopts a generative model to inject noise to the training dataset in order to generate a perturbation-resilient DNN model. While this approach can be effective in achieving the goal of visual privacy protection, it requires a full-fledged retraining of the target DNN model, which is infeasible for the large volumes of proprietary DNN models already commercially deployed today. Our proposed protective perturbation approach also belongs to this category of visual privacy protection using image blurring. However, our approach addresses the limitations of these state-of-the-art methods by blurring all the features and eliminating the need of re-training the target DNN model. These benefits are achieved by the proposed perturbation generator that leverages the distinction between human and machine visions.

6.2 Adversarial Attacks

Adversarial attacks, such as PGD [27] and C&W [12], have been studied in the deep learning and security communities recently. There have been two major use cases of adversarial attacks presented in the community.

Traditional Adversarial Attacks. In traditional adversarial attacks, very small perturbations (i.e., almost invisible to the human visual perception) added to the input image could significantly alter the DNN inference results (i.e., misleading the machine vision), since the human visual perception is different from the machine vision [49, 56]. Such attacks have been shown to be exploitable by adversaries to compromise the robustness and security of deep learning applications and systems [25, 57], which have drawn a great deal of attention in the deep learning community to develop effective countermeasures [29, 41].

Benign Applications of Adversarial Attacks. Inspired by the traditional adversarial attacks, there have been a small number of recent research efforts that leverage adversarial perturbations as a defense mechanism to prevent potential security attacks on the user's image, audio, or video data [24, 47, 55]. These approaches proactively add adversarial perturbation to the objects (e.g., video and audio) under protection, which would mislead machine learning-based threat models (e.g., face authentication attack) but maintain the original functionality of the application intended for human vision. Different from the existing research, our proposed protective perturbation approach targets an opposite optimization direction, which maximizes the perturbations to interfere with the human vision for privacy protection but minimizes the impact to machine vision (i.e., DNN-based image recognition). The generated perturbations serve as a privacy mask to prevent the exposure of user's private information.

7 LIMITATIONS AND DISCUSSIONS

Although our protective perturbation generator could produce protected images with high effectiveness and efficiency, there is still room for improvement in the current method. In this section, we discuss the technical limitations of the proposed approach, which inspire our future work along this research direction.

7.1 Auxiliary Model Selection

As shown in Section 4, the auxiliary models play an important role in the protective perturbations generation. In our current work, we have evaluated all the target-auxiliary model pairs (i.e., 7 auxiliary models for each target model), which shows varying accuracy results (as shown in Figure 7). The variations indicate potential correlations between the two models based on their similarities or other conditions. In our future work, we plan to conduct a more intensive study on such correlations and further research on an automatic selection mechanism to determine the optimal auxiliary model for a given target model.

7.2 Parameter Tuning

In our current system implementation and evaluation, we mostly fixed the parameters for all the generators with different target and auxiliary models. Given the distinctions between different models, there still exists room for improvement in the performance of the perturbation generator if the parameter tuning process is more customized to the individual models. In the future work, we plan to apply individually tuned parameters for each target model during the training process to further improve the effectiveness. Such customized tuning can also benefit from the automatic auxiliary model selection mechanism (discussed in Section 7.1), as the scope of tuning can be narrowed down to the selected auxiliary models.

7.3 Privacy Evaluation

In our current evaluation, we employ SSIM to calculate the similarity between the protected images and the original images, since it is one of most widely used metrics to evaluate human-perceivable image qualities. However, we acknowledge that there have been studies showing that the SSIM metric may not always reflect human visual perception [30]. Therefore, we plan to apply different

similarity functions in the future experiments, such as Singular Value Decomposition (SVD) [16, 22]. Also, we plan to conduct a user study to further evaluate the effectiveness of the protective perturbations as a privacy protection measure in our future work.

7.4 Evaluation Datasets

We adopt CIFAR-10 as the dataset for evaluation in the current study, as it is a commonly used image datasets to train the image classification models. Also, with CIFAR-10 it is relatively fast to train and run the DNN models while providing a moderate scale of image size (32×32), number of images (50K training images and 10K test images), and number of image classes (10 classes). This facilitates our efforts of implementations and evaluations in the proof-of-concept phase of this study. However, we acknowledge that more comprehensive evaluations with larger datasets are desirable for the deployment of the proposed approach. In our future work, we plan to further evaluate our proposed protective perturbation approach using more datasets with a larger variety of classes and image sizes (e.g., CIFAR-100 [23] and ImageNet [14]).

8 CONCLUSION AND FUTURE WORK

We have developed a protective perturbation generator to preserve user privacy in deep learning-based mobile image recognition applications. The perturbation generator can effectively obfuscate the input image to prevent privacy leakage without impacting the prediction accuracy on the target image recognition model. We evaluated the effectiveness and efficiency of the generator using the CIFAR-10 dataset with 8 target models. The results indicated that the protected images achieved high prediction accuracy on the target models and low prediction accuracy on the auxiliary models representing adversaries. Also, the perturbation generator achieved premium timing performance to support real-time applications. More importantly, the proposed privacy protection approach does not require re-training the already deployed legacy DNN models on the service provider end, making it immediately deployable to benefit real-world systems and applications. The repository of the project is at https://github.com/hwsel/ProtectivePerturbation.

In the future work, in addition to addressing the limitations presented in Section 7, we plan to promote the proposed protective perturbation approach to a broader range of security and privacy sensitive applications, which may involve more advanced DNN or computer vision computations other than classification (e.g., image/video analytics, optical flow, and superresolution), or more advanced multimedia types other than 2D images (e.g., 3D point cloud and 360-degree videos). In this extended set of DNN and multimedia applications, visual privacy is likely to become an even more significant concern given the more prevalent application domains and the finer-granularity of information exposure with the rich multimedia. Meanwhile, it would be more challenging to adapt the protective perturbation generation to these complex scenarios.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive feedback. This work was partially supported by the National Science Foundation under awards CNS-1912593 and CNS-2114220.

REFERENCES

[1] 2005. ARM Security Technology: Building a Secure System using TrustZone Technology.
[2] 2016. GDPR. Intersof Consulting. https://gdpr-info.eu.
[3] 2017. Find It On eBay: Using Pictures Instead of Words. https://tech.ebayinc.com/product/find-it-on-ebay-using-pictures-instead-of-words/.
[4] 2017. Google Lens. Search what you see. https://lens.google.com/.
[5] 2021. HIPAA. US Department of Health and Human Services. https://www.hhs.gov/hipaa/index.html.
[6] 2021. Intel Software Guard Extensions (SGX). https://software.intel.com/content/www/us/en/develop/topics/software-guard-extensions.html.
[7] 2021. Pytorch Lightning Metrics. https://pytorch-lightning.readthedocs.io/en/stable/extensions/metrics.html.
[8] 2021. Pytorch Mobile. https://pytorch.org/mobile/home/.
[9] 2021. Pytorch MS-SSIM. https://github.com/VainF/pytorch-msssim.
[10] 2021. Understand the Intersection between Data Privacy Laws and Cloud Computing. https://legal.thomsonreuters.com/en/insights/articles/understanding-data-privacy-and-cloud-computing.
[11] Rosario Cammarota, Matthias Schunter, Anand Rajan, Fabian Boemer, Ágnes Kiss, Amos Treiber, Christian Weinert, Thomas Schneider, Emmanuel Stapf, Ahmad-Reza Sadeghi, et al. 2020. Trustworthy AI Inference Systems: An Industry Research View. *arXiv:2008.04449*.
[12] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (S&P)*. 39–57.
[13] Jianfeng Chi, Emmanuel Owusu, Xuwang Yin, Tong Yu, William Chan, Patrick Tague, and Yuan Tian. 2018. Privacy Partitioning: Protecting User Data During the Deep Learning Inference Phase. In *arXiv:1812.02863*. 1–17.
[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A Large-Scale Hierarchical Image Database. In *IEEE conference on computer vision and pattern recognition (CVPR)*. 248–255.
[15] WA Falcon et al. 2019. PyTorch Lightning. 3 (2019). https://github.com/PyTorchLightning/pytorch-lightning.
[16] Liyue Fan. 2019. Practical Image Obfuscation with Provable Privacy. In *IEEE International Conference on Multimedia and Expo (ICME)*. 784–789.
[17] Andy Greenberg. 2014. Hacker Lexicon: What Is End-to-End Encryption? *Wired, November* 25 (2014).
[18] Zhongshu Gu, Heqing Huang, Jialong Zhang, Dong Su, Hani Jamjoom, Ankita Lamba, Dimitrios Pendarakis, and Ian Molloy. 2020. Confidential Inference via Ternary Model Partitioning. In *arXiv:1807.00969*. 1–12.
[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.
[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely Connected Convolutional Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4700–4708.
[21] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. 2020. Secure, Privacy-Preserving and Federated Machine Learning in Medical Imaging. *Nature Machine Intelligence* 2, 6 (2020), 305–311.
[22] Suleyman Serdar Kozat, Ramarathnam Venkatesan, and Mehmet Kivanç Mihçak. 2004. Robust Perceptual Image Hashing via Matrix Invariants. In *International Conference on Image Processing (ICIP)*, Vol. 5. 3443–3446.
[23] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.
[24] Chetan Kumar, Riazat Ryan, and Ming Shao. 2020. Adversary for Social Good: Protecting Familial Privacy through Joint Adversarial Attacks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, Vol. 34. 11304–11311.
[25] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2017. Trojaning attack on neural networks. In *Network and Distributed System Security Symposium (NDSS)*.
[26] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101* (2017).
[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv preprint arXiv:1706.06083* (2017).
[28] Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the Machine-Vision Package of Torch. In *ACM international conference on Multimedia*. 1485–1488.
[29] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On Detecting Adversarial Perturbations. *arXiv preprint arXiv:1702.04267* (2017).
[30] Jim Nilsson and Tomas Akenine-Möller. 2020. Understanding SSIM. *arXiv:2006.13846* (2020).
[31] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2017. Towards a Visual Privacy Advisor: Understanding and Predicting Privacy Risks in Images. In *International conference on computer vision (ICCV)*. 3686–3695.
[32] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich. 2010. SCiFI - A System for Secure Face Identification. In *IEEE Symposium on Security and Privacy (S&P)*. 239–254.
[33] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical Black-Box Attacks against Machine

Learning. In *Asia conference on computer and communications security (AsiaCCS)*. 506–519.
[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8024–8035.
[35] Huy Phan. 2021. PyTorch Models Trained on CIFAR-10 Dataset. https://github.com/huyvnphan/PyTorch_CIFAR10.
[36] Huy Phan, Yi Xie, Siyu Liao, Jie Chen, and Bo Yuan. 2020. CAG: A Real-time Low-cost Enhanced-robustness High-transferability Content-aware Adversarial Attack Generator. In *AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 34. 5412–5419.
[37] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4422–4431.
[38] Zhongzheng Ren, Yong Jae Lee, and Michael S Ryoo. 2018. Learning to Anonymize Faces for Privacy Preserving Action Detection. In *European conference on computer vision (ECCV)*. 620–636.
[39] 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*. 234–241.
[40] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. 2009. Efficient Privacy-Preserving Face Recognition. In *International Conference on Information Security and Cryptology (ICISC)*. 229–244.
[41] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-Gan: Protecting Classifiers against Adversarial Attacks Using Generative Models. *arXiv preprint arXiv:1805.06605* (2018).
[42] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4510–4520.
[43] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).
[44] Qianru Sun, Liqian Ma, Seong Joon Oh, Luc Van Gool, Bernt Schiele, and Mario Fritz. 2018. Natural and Effective Obfuscation by Head Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5050–5059.
[45] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–9.
[46] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2818–2826.
[47] Zhongze Tang, Xianglong Feng, Yi Xie, Huy Phan, Tian Guo, Bo Yuan, and Sheng Wei. 2020. VVSec: Securing Volumetric Video Streaming via Benign Use of Adversarial Perturbation. In *ACM International Conference on Multimedia (MM)*. 3614–3623.
[48] Florian Tramer and Dan Boneh. 2019. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. In *International Conference on Learning Representations (ICLR)*. 1–19.
[49] Haohan Wang, Xindi Wu, Zeyi Huang, and Eric P Xing. 2020. High-Frequency Component Helps Explain the Generalization of Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 8684–8694.
[50] Ji Wang, Jianguo Zhang, Weidong Bao, Xiaomin Zhu, Bokai Cao, and Philip S Yu. 2018. Not Just Privacy: Improving Performance of Private Deep Learning in Mobile Cloud. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2407–2416.
[51] Sen Wang and J Morris Chang. 2020. Privacy-Preserving Image Classification in the Local Setting. *arXiv:2002.03261* (2020).
[52] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612.
[53] Zhou Wang, Alan C Bovik, and Eero P Simoncelli. 2005. Structural Approaches to Image Quality Assessment. *Handbook of Image and Video Processing* 7, 18 (2005).
[54] Hao Wu, Xuejin Tian, Minghao Li, Yunxin Liu, Ganesh Ananthanarayanan, Fengyuan Xu, and Sheng Zhong. 2021. PECAM: Privacy-Enhanced Video Streaming and Analytics via Securely-Reversible Transformation. In *International Conference on Mobile Computing and Networking (MobiCom)*. 229–241.
[55] Zirui Xu, Fuxun Yu, Chenchen Liu, and Xiang Chen. 2019. HAMPER: High-Performance Adaptive Mobile Security Enhancement against Malicious Speech and Image Recognition. In *Asia and South Pacific Design Automation Conference (ASPDAC)*. 512–517.
[56] Shaokai Ye, Sia Huat Tan, Kaidi Xu, Yanzhi Wang, Chenglong Bao, and Kaisheng Ma. 2019. Brain-Inspired Reverse Adversarial Examples. *arXiv:1905.12171* (2019).
[57] Yang Zhao, Xing Hu, Shuangchen Li, Jing Ye, Lei Deng, Yu Ji, Jianyu Xu, Dong Wu, and Yuan Xie. 2019. Memory Trojan Attack on Neural Network Accelerators. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1415–1420.
[58] Bingquan Zhu, Hao Fang, Yanan Sui, and Luming Li. 2020. Deepfakes for Medical Video De-Identification: Privacy Protection and Diagnostic Information Preservation. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*. 414–420.
[59] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. Automating Visual Privacy Protection Using a Smart LED. In *International Conference on Mobile Computing and Networking (MobiCom)*. 329–342.