Protecting Visual Secrets using Adversarial Nets

Nisarg Raval, Ashwin Machanavajjhala, Landon P. Cox
Department of Computer Science, Duke University

Abstract

Protecting visual secrets is an important problem due to the prevalence of cameras that continuously monitor our surroundings. Any viable solution to this problem should also minimize the impact on the utility of applications that use images. In this work, we build on the existing work of *adversarial learning* to design a perturbation mechanism that jointly optimizes privacy and utility objectives. We provide a feasibility study of the proposed mechanism and present ideas on developing a privacy framework based on the adversarial perturbation mechanism.

1. Introduction

With advancements in deep learning, machines are becoming extremely efficient at understanding visual scenes. Many applications capture users' surroundings in the form of images or videos and process them to extract useful information. While the utility of these applications is highly sought after, it is equally important to protect sensitive information that may be leaked through captured images or videos to possibly malicious applications.

Consider a scenario where Alice wants to use a translation app that translates text in the captured images. The app needs to access the camera feed in order to perform text extraction and translation. However, she is worried that sensitive objects accidentally captured by the camera (e.g., a medicine bottle or her credit card) will be disclosed to the app. This raises an important question - *How can Alice enjoy the benefits of an untrusted third-party app without sacrificing her privacy?*

Prior methods of protecting visual secrets use vision algorithms to classify images or objects as sensitive and remove them from the camera feed [11] or they only reveal objects of interest and block everything else [4]. These methods suffer from three major limitations. First, the underlying object detection algorithms are not robust against real-world scenarios such as occlusion and illumination variation. Second, marking the extent of sensitive object in training images is human intensive and error prone. Finally, these methods do not protect against an adversary who can exploit known correlations between sensitive and non-sensitive regions of the image [7, 8].

We propose a privacy mechanism that systematically addresses the above shortcomings. We adapt a game-theoretic notion to design an *adversarial perturbation mechanism* that hides visual secrets in the camera feed without significantly affecting the functionality of the target application. In particular, we formulate this problem as a game between two players – an obfuscator and an attacker. The goal of the attacker is to identify visual secrets in the images perturbed by the obfuscator, while the aim of the obfuscator is to perturb images such that the perturbed images are similar to the original images as well as fool the attacker. With these competing objectives, both players are pitted against each other. As the game progresses, both obfuscator and attacker improve their methods (of detecting secrets and perturbing the image, respectively) until the attacker can no longer identify sensitive information in the perturbed images.

In concurrent work, Edwards *et al.* [2] proposed a similar approach with the goal of learning *fair representations* of the data that minimizes an adversary's ability to infer sensitive information. However, they do not explicitly quantify privacy or utility guarantees achieved by the mechanism. The primary focus of our work is to empirically quantify these guarantees. Moreover, we also discuss whether the adversarial perturbation approach can become a basis for formally defining privacy in images.

To summarize, we make the following contributions in this paper. First, we build upon the *adversarial nets* [3] framework to design a perturbation mechanism that jointly optimizes both privacy and utility objectives. Then, we provide a feasibility study on a real-world dataset demonstrating and quantifying the effectiveness of the perturbation mechanism in hiding sensitive information while preserving utility. We end with a discussion of whether privacy in images can be formalized via adversarial nets.

2. Adversarial Perturbation Mechanism

We model the perturbation mechanism using the *adversarial nets* framework. It consists of two competing networks – 1) an *attacker network* A that learns to identify secrets in the obfuscated images, and 2) an *obfuscator network* O that attempts to fool the attacker by learning the correct transformation. We model the privacy and the utility requirements as optimization objectives of these networks such that at the end of training the obfuscated images satisfies both privacy and utility requirements.

Let X represents the universe of all images. $X_S \subseteq X$ is the set of images with some sensitive properties (private images) and $X_U = X \setminus X_S$ is the set of public images. The obfuscator O takes as input an image $x \in X$ and outputs an obfuscated image $O(x, \theta_O) \in X$, where θ_O denotes the parameters of O. One way to ensure that obfuscated images satisfy privacy requirements is by ensuring that the distribution of public images and the distribution of obfuscated images are indistinguishable. We enforce this property by defining an attacker A that attempts to distinguish between these two distributions. In particular, A takes as input an image $x \in X$ and outputs a score $A(x, \theta_A)$ denoting the probability of image x having secrets, where θ_A denotes the parameters of A. For a specific obfuscator O, we find the optimal attacker A by optimizing the following:

$$\begin{aligned}\arg\max_{\theta_A} \mathbb{E}_{x \in X_S} A(x, \theta_A) + \mathbb{E}_{x \in X_U}(1 - A(x, \theta_A)) \\ + \mathbb{E}_{x \in X_S} A(O(x, \theta_O), \theta_A) \\ + \mathbb{E}_{x \in X_U}(1 - A(O(x, \theta_O), \theta_A))\end{aligned} \tag{1}$$

The attacker that maximizes Equation (1) gives high scores to private images and low scores to public images, irrespective of whether they are perturbed or not. In other words, $A(O(x), \theta_A)$ can be interpreted as a privacy loss of the obfuscator O. Thus, we train the obfuscator to minimize this privacy loss by optimizing the following:

$$\begin{aligned}\arg\min_{\theta_O} \lambda \, \mathbb{E}_{x \in X} A(O(x, \theta_O), \theta_A) \\ + (1 - \lambda) \, \mathbb{E}_{x \in X} L_U(x, O(x, \theta_O))\end{aligned} \tag{2}$$

Here, L_U is a utility loss that measures the error in obfuscation. L_U can be any error metric based on the user specified utility requirements. In this work, we use L_1 distance as the error metric for utility loss, i.e., $L_U(x, O(x, \theta_O)) = ||x - O(x, \theta_O)||_1$. This loss enforces the obfuscated images to be similar to the original images (utility requirement) while the first term enforces the obfuscated images to be indistinguishable from public images (privacy requirement). The trade-off between privacy and utility can be controlled by the λ parameter.

2.1. Network Architecture and Training

Our network architecture is inspired by the deep convolutional generative adversarial networks (DCGANs) [9]. The attacker network takes as input a color image. The input layer is followed by four convolutional layers each with a ReLU (rectified linear unit) activation. Each convolutional layer uses filters of size 5×5 with stride 2. The number of filters in each layer are 64, 128, 256 and 512, respectively. The final layer is a fully connected layer with a single unit. We use sigmoid activation in the last layer to bring the output in [0,1] range.

The obfuscator network is a denoising autoencoder. The goal of a denoising autoencoder is to reconstruct a clean input from the noisy input. We can think of the secrets as the noise that we would like to remove from the input image. Specifically, the obfuscator takes as input a color image. The input layer is followed by two convolutional layers with a ReLU activation. The number of filters in each layer are 64 and 128, respectively. Each filter is of size 5×5 with stride 2. This is followed by a fully connected layer with 8192 units and ReLU activation. The output of a fully connected layer is upsampled by four deconvolutional layers. The first three layers have 512, 256 and 128 filters each followed by ReLU activation. The final layer uses one filter followed by $tanh$ activation to produce a color image. Each convolutional layer uses filters of size 5×5 with stride 2.

We train both networks simultaneously using an iterative algorithm proposed by Goodfellow *et al.* [3]. On every iteration (epoch) we update parameters of both the networks as follows. First, we use the obfuscator to generate perturbed images using private as well as public training images. Then, we use both original (unperturbed) and perturbed images to train the attacker. Finally, we use loss from the attacker and utility function to update the obfuscator. Initially, the obfuscator will either fail to hide the secret (high privacy loss) or generate an image that is completely different from the input image (high utility loss). After several epochs, it will adjust its parameters to successfully hide secrets from the attacker. The attacker will soon catch up by learning the obfuscation process which will force the obfuscator to develop a better perturbation mechanism. Eventually, the obfuscator learns a correct perturbation such that the attacker no longer differentiates between a perturbed image with a secret and an image without a secret.

3. Experiments

We quantify the privacy achieved by our mechanism and its impact on the utility on the image classification dataset CIFAR10 [5]. It consists of 32×32 color images from 10 categories. We selected two categories – horse and airplane, for our experiments. Thus, we have 10,000 training images and 2,000 test images, which together form the set of *public images* (X_U). Using these images we generated a set of *private images* (X_S) as follows. For each image in X_U, we generated a new image by placing a random QR code [1] of size 10×10 at a randomly chosen location in the image. The experiments evaluate the mechanism's ability to hide QR codes while preserving the overall structure of the image. We trained both the networks using the Adam

[1] generated using qrcode library - https://pypi.python.org/pypi/qrcode

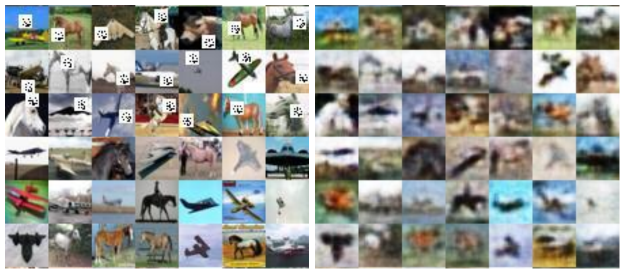
(a) Input images (b) Perturbed images

Figure 1: Perturbation results on private and public images.

optimizer with a learning rate of 0.0001 for 1,000 epochs and a batch size of 50. Both networks are implemented in Tensorflow [2] and are trained on NVIDIA Tesla K80.

3.1. Qualitative Results

Figure 1 shows randomly sampled test images together with their perturbation results. The first three rows are private images while the last three rows are public images. We can see that the obfuscator successfully removes QR codes from all the private images. Moreover, it preserves the overall shape and color profile of both public and private images. Note that during evaluation, the obfuscator only gets an image as the input. It does not get an explicit label such as whether the image is public or private. Moreover, we train a single obfuscator that handles both public and private images. Given an image, the obfuscator automatically determines if the input image is public or private and removes any sensitive content if present in the image.

3.2. Utility Evaluation

We quantify the utility of our mechanism in terms of how well it classifies public objects (horse vs airplane) in the perturbed images. For this task, we used a CNN with four convolutional layers followed by two fully connected layers. We trained the network for 100 epochs with a batch size of 50. When trained on original (unperturbed) images, CNN achieved classification accuracy of 96% on original test images. We evaluate the accuracy on perturbed images against this benchmark accuracy under two training scenarios – a) without access to perturbed images, and b) with access to perturbed images. For the first case we used CNN trained on original images to classify perturbed images which resulted into 86% accuracy. For the second case we retrained CNN with available perturbed training images which improved the classification accuracy to 91%. These results demonstrate that the perturbation mechanism does not significantly impact the utility and preserves the properties of public objects. Thus, it can be used in a scenario described in Section 1 to hide sensitive information without affecting the functionality of the text translation app.

[2]https://www.tensorflow.org/

3.3. Privacy Evaluation

We quantify privacy in terms of 1) how well an adversary can detect whether the perturbed image had QR code or not, and 2) how well the adversary can detect position of the QR code in the perturbed image.

For the first task we used CNN based binary classifier to classify perturbed images into private (with QR code) and public (without QR code). We consider two types of adversaries – a *weak adversary* and a *strong adversary*. The weak adversary does not have access to the perturbation mechanism and hence can only use original training images to train the CNN. We found that weak adversary achieved 50% classification accuracy. In other words it is as good as randomly guessing whether the perturbed image was private or public. Thus, without the access to the perturbation mechanism, the attacker can not perform better than random guessing. The strong adversary has a black box access to the perturbation mechanism which it can use to generate perturbed images with label (public or private) for training. We found that training the CNN with perturbed images improved the adversary's accuracy to 75%. Thus, even with the knowledge of the perturbation mechanism adversary had a significant error rate (25%).

For the second task, we used template matching to find the position of QR code in perturbed images. Given a perturbed image we used a corresponding QR code as a template and performed matching using normalized correlation coefficient method. We say a QR code is found if the overlap between the predicted bounding box and the true bounding box is more than 50%. In none of the perturbed images, we were able to find QR codes which shows that the perturbation mechanism successfully hides visual secrets. Thus, the proposed mechanism successfully hides the sensitive content and limits the adversary's ability to learn whether or not a secret is present in the perturbed image.

4. Discussion

In this section, we briefly outline key challenges of designing privacy methods using deep learning and describe how our mechanism addresses these challenges.

Deep nets as adversaries: Given that *deep learning* is extremely successful in understanding images, it seems the right tool to address privacy issues in vision. Researchers have used deep learning to exploit vulnerabilities in the existing methods of image obfuscation [7, 8]. Moreover, deep learning based obfuscation techniques have been proposed to hide sensitive information in the data [1, 2]. Neural networks are *universal approximators* [10] which make them natural candidates for simulating various classes of adversaries with different capabilities. It is an interesting future direction to use this approach to design rigorous privacy definitions that provide formal privacy guarantees against such machine learning based adversaries.

Correlation attacks: Image data is rich in contextual information which makes it vulnerable to correlation attacks. For instance, simply blocking faces is not enough to conceal users' identity because seemingly innocuous contextual information such as clothes, gait, etc. could reveal the true identity [8]. We believe that the proposed adversarial perturbation mechanism will protect against such correlation attacks. Intuitively, if the obfuscator network fails to hide such correlations then the attacker network can exploit them to distinguish private images from public images. We plan to investigate this further through empirical analysis.

Post-processing: We say that a privacy mechanism $\mathcal{M}$ satisfies the post-processing requirement with respect to some algorithm $\mathcal{A}$, if the composite mechanism $\mathcal{A} \circ \mathcal{M}$ provides the same privacy guarantees as provided by $\mathcal{M}$ [6]. For instance, a mechanism *Blur* that protects the identity of individuals by blurring faces does not satisfy this property. One can simply design a de-blurring algorithm A that negates the blurring effect revealing the true identity of people. Our hypothesis is that the adversarial perturbation mechanism satisfies post-processing requirement under certain conditions. Here, we provide an informal argument and leave the formal proof for future work. The formulation of our optimization problem forces the obfuscator to minimize privacy loss against worst-case adversary A among all possible adversaries in domain $\mathbb{A}$. Hence, in principle, we can bound the expected privacy loss of the obfuscator O via the expected success rate of the worst-case attacker A. In other words, one can not find an adversary $A' \in \mathbb{A}$ that is (in expectation) strictly better than A simply because A is the worst-case adversary with respect to the obfuscator O.

Privacy without obscurity: Algorithms that rely on obscurity, or the fact that the algorithm or its parameters are secret to achieve security are prone to various attacks. Thus, it is important for a privacy mechanism to hide sensitive information without relying on the principle of obscurity. The *Blur* mechanism mentioned above is prone to information leak even if the underlying blurring algorithm (or parameters) is kept secret [7]. The adversarial perturbation mechanism proposed in this work uses a deterministic transformation to remove sensitive information from the image. Hence, it can leak information if the adversary knows the exact structure and parameters of the obfuscator network O. For example, given a private image I and an obfuscated image I', the adversary can verify if I' is the obfuscation of I by simply passing it through O. Abadi *et al.* [1] address this problem in the context of cryptography by providing a randomized private key as an additional input to the network. We plan to investigate if similar randomization can can help our mechanism not be susceptible to such attacks.

Training data: One of the limiting factors of using neural networks to build a privacy framework is the need for large amount of labeled training data. We partially address this issue by removing the requirement of labeling objects in training images. Our framework only requires image level labels such as whether the image is public or private. However, getting private images could be challenging in many scenarios. Furthermore, it is not clear how the mechanism will behave if given an image with a sensitive object that it has never seen. One can address this issue using the concept of one-shot learning [12] which is an active area of research.

5. Conclusions

We model the problem of protecting visual secrets as an adversarial game between an obfuscator and an attacker with the competing goals of protecting and revealing secrets, respectively. We quantified the privacy leak and utility loss of our mechanism through empirical evaluation. We also discussed various challenges in establishing the proposed adversarial approach as basis for privacy in vision.

References

[1] M. Abadi and D. G. Andersen. Learning to protect communications with adversarial neural cryptography. *CoRR*, 2016.

[2] H. Edwards and A. J. Storkey. Censoring representations with an adversary. *CoRR*, 2015.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*. 2014.

[4] S. Jana, D. Molnar, A. Moshchuk, A. Dunn, B. Livshits, H. J. Wang, and E. Ofek. Enabling Fine-Grained Permissions for Augmented Reality Applications With Recognizers. In *USENIX Security*, 2013.

[5] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Master's thesis, 2009.

[6] A. Machanavajjhala and D. Kifer. Designing statistical privacy for your data. *Commun. ACM*, 2015.

[7] R. McPherson, R. Shokri, and V. Shmatikov. Defeating image obfuscation with deep learning. *CoRR*, 2016.

[8] S. J. Oh, R. Benenson, M. Fritz, and B. Schiele. Faceless person recognition; privacy implications in social media. *CoRR*, 2016.

[9] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, 2015.

[10] S. Sonoda and N. Murata. Neural network with unbounded activations is universal approximator. *CoRR*, 2015.

[11] R. Templeman, M. Korayem, D. Crandall, and A. Kapadia. PlaceAvoider: Steering first-person cameras away from sensitive spaces. In *NDSS*, 2014.

[12] O. Vinyals, C. Blundell, T. P. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. *CoRR*, 2016.