

AN ABSTRACT OF THE DISSERTATION OF

Arezoo Rajabi for the degree of Doctor of Philosophy in Computer Science
presented on February 25, 2021.

Title: Two Sides of a Coin: Adversarial-Based Image Privacy and Defending
Against Adversarial Perturbations for Robust CNNs

Abstract approved: _____

Rakesh B. Bobba

Emergence of highly accurate Convolutional Neural Networks (CNNs) with the capability to process large datasets, has led to their popularity in many applications, including safety/security-sensitive (*e.g.*, disease recognition, self-driving cars). Despite the high accuracy of convolutional neural networks, they have been found to be susceptible to adversarial noise added to benign examples and out-distribution samples that are classified confidently into in-distribution classes. The applications of CNNs in surveillance services necessitate the need for secure and robust CNNs. On the other hand, despite the benefits of CNNs to surveillance applications, they pose a privacy threat as they are able to undertake image face recognition on a large scale. Coupled with the availability of large image datasets on online social networks and at image storage providers, this poses a serious pri-

vacy threat. Emergence of Super Resolution Convolutional Neural Networks (SRCNNs) which improve the image resolution for face recognition classifiers further exacerbates this threat.

In this dissertation, we address both these problems. We first propose taking advantage of CNNs vulnerability to adversarial perturbations by adding adversarial noise to images to fool CNNs to protect privacy of images in cloud image storage setting. We propose and evaluate two adversarial-based protection methods: (i) a semantic perturbation-based method called, *k*-Randomized Transparent Image Overlays (*k-RTIO*), and (ii) a learning-based method called, Universal Ensemble Perturbation (UEP). These methods can thwart unknown face recognition models (*i.e.*, black-box) while requiring low computational resources. We then evaluate the practicality of adversarial perturbations learned for CNNs on SRCNNs and show that adversarial perturbations are transparent to SRCNNs.

In the last part of our dissertation, We propose mechanisms to make CNNs robust against adversarial and out-distribution examples by rejecting suspicious inputs. In particular, we propose an Augmented CNN (A-CNN) with an extra class that is trained on limited out-distribution samples, which can improve CNNs resiliency against adversarial examples. Further, to protect pre-trained highly accurate CNNs, we propose using *adversarial profiles*, perturbations that misclassify samples of a source class (not other classes) to a target class, as a post-processing step to detect out-distribution examples.

©Copyright by Arezoo Rajabi
February 25, 2021
All Rights Reserved

Two Sides of a Coin: Adversarial-Based Image Privacy and Defending
Against Adversarial Perturbations for Robust CNNs

by

Arezoo Rajabi

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented February 25, 2021

Commencement June 2021

Doctor of Philosophy dissertation of Arezoo Rajabi presented on
February 25, 2021.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Arezoo Rajabi, Author

ACKNOWLEDGEMENTS

I would first like to thank my supervisor, Professor Rakesh B. Bobba, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I also would like to thank my PhD committee members, Dr. Jinsub Kim, Dr. Xiaoli Fern and Dr. Mike Rosulek for all of their invaluable feedback, support, and for selflessly agreeing to serve on my committee.

During my years working at Oregon State University, it has been a true pleasure to collaborate with Dr. Mahdiah Abbasi, Dr. Charles Wright, Dr. Wuchi Feng and Dr. Mike Rosulek. Especially, I am truly appreciative of my dear friend Mahdiah Abbasi for all helpful discussions, suggestion, and collaboration.

In addition, I could not have completed this dissertation without the support of my friends, Hamidreza Zoraghein, Parisa Ataei, Mahtab Aboufazeli, Mahdiah Abbasi, and Mojgan Hatami who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Finally, I must express my very profound gratitude to my parents, Ali Rajabi and Hamideh Ghourchibeigi, and my sisters, Roya and Mojgan, for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them and their sacrifices.

TABLE OF CONTENTS

	<u>Page</u>
1 Introduction	1
2 Preliminaries and Background	6
2.1 Convolutional Neural Network Classifiers	6
2.2 Super Resolution Convolutional Neural Networks (SRCNNs)	7
2.3 Adversarial Examples and Perturbations	8
3 Adversarial Perturbation as Image Privacy Defense	11
3.1 Introduction	11
3.2 Related Work	15
3.3 System Model and Requirements	18
3.4 Preliminaries	20
3.5 Proposed Approaches	22
3.5.1 Universal Ensemble Perturbation	23
3.5.2 k-Randomized Transparent Image Overlays	28
3.6 Evaluation	33
3.6.1 UEP Performance	34
3.6.2 k-RTIO Performance	39
3.6.3 Potential Attacks Against UEP and k-RTIO	46
3.7 Discussion and Future Work	51
3.8 Conclusion	54
4 Adversarial Examples against Super Resolution Convolutional Neural Networks	56
4.1 Introduction	56
4.2 System and Adversary Model	59
4.3 Related Work	61
4.4 Methodology	62
4.4.1 Overview	62
4.4.2 Generating Low Resolution Adversarial Images	63
4.4.3 Survivability Evaluation	64

TABLE OF CONTENTS (Continued)

	<u>Page</u>
4.4.4 Metrics	67
4.5 Evaluation	69
4.5.1 Simulation Setup	70
4.5.2 Adversarial Perturbations	71
4.5.3 Adversarial Examples	76
4.6 Future Work	83
4.7 Conclusion	84
5 Detection and Rejection of Adversarial Examples and Out-Distribution Samples	85
5.1 Introduction	85
5.2 Related Work	88
5.3 Augmented Convolutional Neural Network	91
5.4 Evaluation	93
5.5 Adversarial Profile for Out-Distribution Samples Detection	99
5.5.1 Evaluation	102
5.6 Future Work	105
5.7 Conclusion	106
6 Conclusion	107
Bibliography	109
Appendices	124
A UEP Setups	125
B k-RTIO Setups and Extra Evaluation	128

LIST OF FIGURES

Figure	Page
2.1 VGG-16 Convolutional Neural Network [14].	6
3.1 From left to right: original image, UEP perturbed image, and k-RTIO perturbed image.	12
3.2 From bottom to top: The last row shows the original images of celebrities. The middle row represents the UEP perturbed images. The first row displays the recovered images.	24
3.3 UEP method learns a universal perturbation δ on a few local CNNs (Y_1, \dots, Y_c) for set of given inputs x_1, \dots, x_n	25
3.4 k-RTIO Scheme. ID of main image is used to select k overlay images from the set S and then ID of selected images are used to generate a permutation. Both overlay selection and block permutation algorithm use the user's secret key.	28
3.5 Left to right: original, k-RTIO perturbed and recovered images.	29
3.6 UEP recognizability vs. β that controls noise level	32
3.7 Pseudorandom method for creating overlay images with permuted blocks for an image with identifier id . # of blocks per overlay image is b . Set of candidate overlay images is S	33
3.8 Using a convolutional layer with stride value of $(2, 2)$ and kernel values of 0.25, one can implement a thumbnail function to reduce the size of an image to 25% of original size.	34
3.9 UEP performance on Google Vision API face detection model, and DeepFace face detector and recognition models.	36
3.10 Google Vision API face detection model performance on kRTIO images for $k=3$ and $k=8$	41
3.11 Clarifai.com (online) and DeepFace (offline) face recognition and detection models on k-RTIO perturbed images for $k = 3$	43
3.12 Different α values vs. size of block for $k = 3$ overlay images. Overlay images were chosen randomly but the same overlay images were used in all cases.	44

LIST OF FIGURES (Continued)

<u>Figure</u>	<u>Page</u>
3.13 Different number of overlay images vs. different number of block sizes for $\alpha = 0.6$. Used same overlay images per row.	45
3.14 SSIM histogram for recovered UEP and k-RTIO images. In contrast to k-RTIO, UEP only perturbs faces in an image, therefore it has larger SSIM on average.	46
3.15 UEP perturbation estimation and removal. The first image is the perturbed image, the second one is median of 200 perturbed images. The third image is inverted median and the final image is the recovered image by an adversary ($\lambda = 0.42$).	47
3.16 Target CNN can recover 60% of the images by using Perturbation Estimation and Removal method just with 5 images using the same perturbation. By setting λ value to 0.48, an adversary can improve its success rate to 70%.	48
3.17 The accuracy of the CNN of TinyImageNet on perturbed images by k-RTIO images for $k = 3$ (left) for the original CNN learned on benign images, and (right) robust CNN which is learned on the k-RTIO images.	49
3.18 Filtering results on DeepFace face detection and celebrity recognition models for $\alpha = 0.45$ and $k = 3$	50
4.1 An adversary uses an SRCNN to generate HR images and gives it to an image CNN classifier for classification. (a) shows an ideal system in which the adversary's SRCNN is known and a user can add adversarial noise to generated HR by the SRCNN, (b) shows an real system in which the adversary's SRCNN is unknown and a user perturbs her LR images directly.	57
4.2 Two different approaches for generating low resolution adversarial examples for unknown SRCNN(s) using only a local CNN(s). In (a), an adversarial example is generated on a clean HR image and the down-scaled adversarial image is given to SRCNN(s) as input. In (b), an adversarial example is generated on an LR clean image and is given to SRCNN(s) as input. Target CNN(s) may or may not be known for users, but the target SRCNN(s) is unknown.	58

LIST OF FIGURES (Continued)

<u>Figure</u>	<u>Page</u>
4.3 Two approaches of selecting a reference image for similarity metrics: (i) The original HR adversarial image, and (ii) Directly perturbed generated HR image.	65
4.4 Each column shows the density distributions for a similarity metric to compare (i) the generated high resolution images from down-scaled clean images and original HR images ($\text{Sim}(I, hr_k(T(I)))$), (ii) the generated high resolution images from down-scaled UEP perturbed images and original UEP perturbed images ($\text{Sim}(I + \delta, hr_k(T(I + \delta)))$), and (iii) the generated high resolution images from down-scaled UEP perturbed images and the directly UEP perturbed high resolution images generated from down-scaled clean images ($\text{Sim}(I + \delta, hr_k(T(I)) + \delta)$). Sim function is a similarity function, PSNR, SSIM or perceptual similarity (PerSim). Two more similar images return higher values for PSNR and SSIM and lower value for perceptual similarity.	72
4.5 The first row shows the low resolution of the second row images and the third row shows generated HR images of low resolution images shown in the first row by CAR(x4). The first column of images shows the clean images and the rest column shows the adversarial examples generated for benign high resolution images with FGS attack and different ϵ values. I , δ , $T(\cdot)$ and $hr(\cdot)$ are the original image, adversarial noise, down-scaling function and SRCNN function which returns high resolution of a given images, respectively. . . .	76
4.6 The first column shows a clean low resolution image and its high resolution images by CAR (x4) and the rest columns show the adversarial examples generated on low resolution images and their high resolution images by CAR (x4). As shown by increasing the ϵ values leads the adversarial noise to increase and the quality of generated high resolution images to decrease.	82
5.1 Adversarial examples for MNIST (first row) and CIFAR-10 (second row). From left to right: original images, T-FGS, FGS, DeepFool, and C&W (L_2) adversarial examples	86

LIST OF FIGURES (Continued)

<u>Figure</u>	<u>Page</u>
5.2 Two-moons synthetic dataset. (a) MLP trained on only in-distribution samples (b) the augmented MLP trained on both in-distribution and some out-distribution samples	90
5.3 Visualization of some randomly selected test samples and their corresponding adversaries (FGS and T-FGS) in the feature spaces (the penultimate layer) learned by a naive CNN and an augmented CNN. To produce and manipulate the 3D plots refer to https://github.com/mahdaneh/Out-distribution-learning_F_Svisulization	97
5.4 Intra and Inter class transferability matrices. The element at $[i, j]$ in Inter class transferability matrix represents the value of $p_{i,j}$. Similarly, the element at $[i, j]$ in intra class transferability matrix represents the value of $e_{i,j}$. The larger value for $p_{i,j}$ and lower value for $e_{i,j}$ are preferred.	103

LIST OF TABLES

Table	Page
4.1 Evaluation scenarios. We consider two attack models of white-box and black-box attacks with different approaches of generating LR adversarial images.	70
4.2 Accuracy of clarifai.com celebrity face detection and recognition models for generated high resolution images from clean images (Baseline) and generated high resolution images from down-scaled UEP perturbed images.	73
4.3 Transferability of generated HR from down-scaled adversarial examples (TR_R) and transferability of directly perturbed HR images generated from LR clean images (TR_I) for both black-box (ResNet152) and white-box setting (ResNet101)	79
4.4 Similarity metrics for evaluating adversarial examples survivability through SRCNNs. We measure PSNR, SSIM and perceptual similarity for generated HR images from down-scaled of adversarial examples for two different cases of (i) original adversarial examples as reference images ($\text{Sim}(hr(T(I + \delta)), I + \delta)$), and (ii) directly perturbed HR images generated from clean low resolution images ($\text{Sim}(hr(T(I)) + \delta, I + \delta)$) as reference images.	80
4.5 Evaluating the survivability of white-box attacks learned on low resolution images for CAR (x4) SRCNN. The first column shows the ϵ value for FGS attack. The second column shows transferability of high resolution images generated from low resolution adversarial examples and the rest of columns show similarity between the HR generated from clean LR image and the generated HR image from its LR resolution adversarial example.	81
5.1 Performance on black-box adversaries attacks. “Acc.” corresponds to accuracy (the rate of correctly classified samples), “Rej.” is the rejection rate, while “Err.” is the misclassification rate All results reported are percentages (%).	96
5.2 Adversarial and out-distribution detection rates.	104

LIST OF APPENDIX FIGURES

<u>Figure</u>	<u>Page</u>
B.1 Set S of the overlay images used in generating k-RTIO images . . .	128
B.2 Blind Signal Separation can recover perturbed images by transparent overlays technique. Generating overlay images by permuting the original overlay images' blocks randomly leads BSS to not be able to recover images.	130

Chapter 1: Introduction

In the last two decades, access to considerable computational power (*e.g.*, GPUs) and large data resources, as well as the advent of complex models, has led to remarkable improvements in deep neural networks in vision tasks such as image classification and object recognition [47]. For example, Convolutional Neural Networks (CNNs) have been shown to achieve high accuracy on large datasets (*e.g.*, 98% top-5 accuracy on the ImageNet dataset, which includes more than 1M images [102]). In addition to image classification, these deep neural networks can be used to improve image quality and create high resolution images with valuable details from low resolution images [23].

The remarkable performance of CNNs led to the popularity of these tools in many applications, such as safety-sensitive applications (*e.g.*, auto-pilot car or disease recognition). While these tools provide benign and beneficial services to users, they also pose a significant privacy threat. The advent of social networks and cloud as a popular platforms for image sharing and storage respectively led to millions of photographs (especially personal images) to be shared or stored online. One can use this large set of available images to learn highly accurate face recognition and super resolution models to track user relationships or activities [108].

Despite the high accuracy of CNNs, it has been demonstrated that they are highly susceptible to adversarial examples, especially when dealing with high di-

mensional inputs such as images [121; 119; 36; 57; 18; 82]. An adversarial image is one that has been perturbed with a noise signal designed to fool CNNs. Furthermore, another serious challenge for CNN-based systems is that when a test sample comes from a different concept or class that is not part of the training set (*i.e.*, in-distribution samples), then CNNs assign them to one of the predefined classes they are trained on, often with high confidence. Due to the significant role of CNNs in many safety-sensitive applications, it is necessary to make them robust against such inputs. On the other hand, the vulnerability of CNNs to adversarial noise might be useful for preserving privacy in image sharing platforms. Here, we aim to answer following questions:

Is adversarial perturbation practical for preserving image privacy? Adversarial perturbations [35; 36] have the nice property that the perturbed images are perceptually recognizable for humans while they are misclassified by a CNN. Recently adversarial image perturbation has been proposed as a way to preserve image privacy by fooling the CNNs [51; 105]. However, these methods are not practical for the real world due to their unrealistic assumptions, such as having access to the target CNN or needing to create large CNNs to generate adversarial perturbations. In other words, these methods did not consider the practical issues of low computational cost and image recoverability.

Therefore, here we first identify the practical requirements that a perturbation-based privacy-preserving approach should satisfy. Then we propose two novel schemes for adversarial perturbation of images that satisfy the identified requirements and do not require special knowledge of the target CNNs. One scheme is

based on *learned adversarial perturbations* [18], while the other is based on *semantic adversarial perturbations* [48]. In particular, we propose *Universal Ensemble Perturbation (UEP)* approach where we learn a single perturbation that can be applied to multiple images, without requiring special knowledge of the target CNN and with minimal loss in the recovery process. Compared to previous adversarial perturbation learning techniques, our UEP approach uses fewer and smaller local CNNs trained over a smaller training set. Also, we propose *k-Randomized Transparent Image Overlays (k-RTIO)*, an approach that derives many unique image perturbations from a small number of overlay source images and a secret key. This semantic adversarial perturbation approach does not require special knowledge of the target CNN and does not require users to train their own CNNs.

Can adversarial perturbations learned only on CNNs survive through super resolution convolutional neural networks? Super Resolution Convolutional Neural Networks (SRCNNs) were designed to generate/recover a high resolution image from its low resolution image. To the best of our knowledge there is no work that investigated the survivability of adversarial noises or perturbations learned for fooling CNN classifiers through SRCNNs. Here we first define the survivability of adversarial images through SRCNNs, then introduce our methodology and metrics for evaluating the survivability of adversarial examples (white-box) and perturbations (black-box) learned on CNN classifiers. Finally we demonstrate that adversarial noises propagate through SRCNNs and that generated high resolution images from low resolution of adversarial examples and adversarial perturbed images can fool CNNs successfully.

How to reduce CNNs’ misclassification rate? Reducing the misclassification rate of CNNs is critical for developing secure and dependable CNN-based intelligent systems, particularly in hostile environments. It has been shown training CNNs along with adversarial examples can improve the resiliency of CNNs against adversarial examples [74]; however, this approach is not practical for large and complex CNNs [17], and is not able to address out-distribution samples. Accordingly, rejecting suspicious inputs has become a desirable solution (which aligns with our privacy-preserving goal as well). There are two main approaches to reject adversarial and/or out-distribution examples: (i) re-training CNNs on in-distribution samples along with auxiliary data such as adversarial examples (*e.g.*, [74; 64]) or (ii) adding a post-processing step to reject suspicious samples [80; 68]. Most proposed methods following the first approach need to be trained on a wide range of adversarial/out-distribution samples [74]. Here, we show that adding an additional *dustbin* class containing *limited natural out-distribution samples* (*i.e.*, natural examples that are statistically and semantically different from in-distribution samples), and *interpolated in-distribution data* (created by interpolating selected pairs of in-distribution samples from two different classes) can lead to an augmented CNN with lower error rate (*i.e.*, misclassification rate) in the presence of adversarial examples.

There are a lot of highly accurate CNNs for different tasks, and users would prefer to improve these available networks to make them robust to out-distribution samples (by employing the second approach) rather than train new CNNs. Accordingly, we propose to create a set of such targeted adversarial perturbations(

called *adversarial profile*) to detect out-distribution and adversarial samples. More precisely, we propose to learn a set of adversarial perturbations (adversarial profile) for each in-distribution class and leverage such profiles to identify and reject adversarial samples.

The rest of the dissertation is organized as follows: In Chapter 2 we present some preliminaries on CNNs, SRCNNs and adversarial attacks. We introduce our adversarial learning based approach, Universal Ensemble Perturbation (UEP) and our semantic adversarial perturbation approach, k Randomized Transparent Overlays (k-RTIO) approach in Chapter 3. We evaluate the survivability of adversarial images learned to fool CNN classifiers through SRCNNs. In Chapter 5, we introduce our proposed methods for adversarial and out-distribution examples detection. Finally we summarise our achievements in Chapter 6.

Chapter 2: Preliminaries and Background

In this chapter, we first introduce Convolutional Neural Network (CNN) classifiers, Super-Resolution Convolutional Neural Networks (SRCNN) and then we describe methods for learning adversarial examples and perturbations.

2.1 Convolutional Neural Network Classifiers

Convolutional Neural Networks (CNNs) as modern deep learning models are able to achieve nearly human-level performance on several computer vision tasks such as face detection [122], optical character recognition [37], object recognition [120], and object detection [101; 99]. A CNN is a series of convolution, pooling, non-

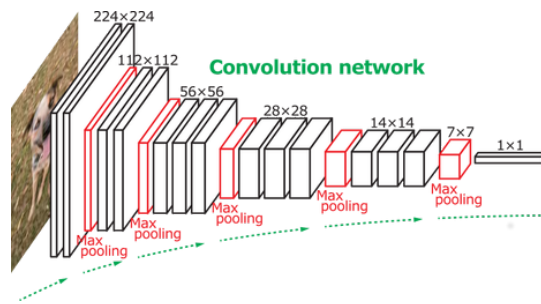


Figure 2.1: VGG-16 Convolutional Neural Network [14].

linear activation layers that are followed by a few fully connected layers. For a K -class classification task, usually a CNN is terminated by a softmax activation layer to map the *logit* layer (with K outputs) into conditional class probabili-

ties. Concretely, a CNN can be denoted by a function $F(·; \theta) : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X} = R^{D^1}$ and $\mathcal{Y} = [0, 1]^K$ are input and output spaces, respectively, with θ compactly denoting parameters of the CNN. Feeding the CNN (i.e. $F(·, \theta)$) with an input sample $x \in R^D$, it returns a vector of conditional class probabilities over K classes s.t. $\sum_{k=1}^K F_k(x; \theta) = 1$. To learn the parameter θ , cross-entropy loss function $J(·, ·; ·) : (\mathcal{X} \times \mathcal{Y}) \rightarrow R^+$ is minimized over N i.i.d. training samples, i.e. $(x^i, t^i)_{i=1}^N$, where $t^i \in \{0, 1\}^K$, as the true class associated with i -th input, is a binary vector with a single one at its k -th element. Formally, the cross-entropy is defined for a given sample (x, t) ² as follows:

$$J(x, t; \theta) = \sum_{k=1}^K t_k \log F_k(x; \theta) \quad (2.1)$$

2.2 Super Resolution Convolutional Neural Networks (SRCNNs)

Super Resolution Convolutional Neural Networks (SRCNNs) are deep convolutional neural networks were proposed to generate high Resolution (HR) images from their low resolution (LR) counterpart [136; 73; 115] with the aim of increasing the visual quality of HR images. In general, SRCNNs optimize Peak Signal to Noise Ratio (PSNR), Minimum Square Error (MSE) or Structural Similarity Index (SSIM) [138] between the original high resolution images and the reconstructed HR images as a objective functions. Here we use $hr(\cdot)$ to denote an SRCNN which takes low resolution of image I (I_{lr}) and generates a high resolution image of it.

¹For image samples, usually the input space is $[0, 1]^D$

²For simplicity in notation, the superscript i is dropped from (x^i, t^i)

2.3 Adversarial Examples and Perturbations

Adversarial generation methods find and add a small adversarial noise (δ) to an image that will cause the target CNN to misclassify it.

$$\begin{aligned} \min_{\delta} \|\delta\|_2 + J(F(x, \theta)) \\ \text{s.t. } \operatorname{argmax} F(x + \delta) \neq y^*, \quad x + \delta \in [0, 1]^d \end{aligned} \tag{2.2}$$

where δ , x , y^* and $J(\cdot)$ are the adversarial noise, given input image, true label of input image, and the loss function, respectively. Since this noise is imperceptible to human eyes, benign images with adversarial noise called *adversarial examples*. Many methods have been proposed to generate a small adversarial noise for a known CNN (e.g., [36; 82; 18; 121]), but they suffer from low *transferability* to unknown CNNs. Therefore, transferable perturbation generation methods were proposed to address this issue. Transferable adversarial perturbations are visible, but the perturbed images with adversarial perturbation are still recognizable for human eyes, but can fool any CNN classifiers with high probability.

Fast Gradient Sign (FGS) [36]: This method aims to minimize the maximum changes in each pixel for a untargeted attack. More precisely, it tends to minimize L_∞ . Therefore, it uses the sign of gradient of loss function as follows:

$$x_{new} = x + \epsilon \cdot \operatorname{sign}(\nabla J(F(x, \theta), y^*)) \tag{2.3}$$

In that, y^* is the true label and ϵ is a constant small coefficient which controls the maximum changes of pixels.

Deepfooling [82]: Unlike the fast gradient sign method, this method is for untargeted attack. This method is a fast and efficient method to find a small perturbation for a given image. Similar to L-BFGS, it minimizes $\|\delta\|_2$. This method is based on an iterative linearization of the loss function.

Carlini and Wagner (C&W) Attack [18]: In contrast to previous methods, this method does not use loss function to find a perturbation. Authors of [18] introduced a new objective function based on minimum difference between the probabilities assigned to the true class and other classes such that the objective function has lower value if a CNN misclassifies the image with higher confidence as follows:

$$f(x) = \max(\max\{Y(x)_j : j \neq y^*\} - Y(x)_{y^*}, -\kappa) \quad (2.4)$$

where $Y(x)$ is the probability vector assigned by CNN to the input(x). Also, y^* and κ denote the real label of the input and confidence parameter, respectively. Larger value for confidence parameter leads the classifiers to assign the image to a new class different from the true one with higher probability such that the difference between the probability of the new class and others is κ at least. Moreover, previous methods add perturbation directly to images. Therefore, finding a perturbation that keeps the final value of images in acceptable range $[0, 1]$ is difficult. To simplify

the problem, they suggested using the alternative term of $\frac{1}{2}(\tanh(z) + 1)$ whose value is always in $[0, 1]$. More precisely, the perturbation δ is added to arc-tangent hyperbolic value of an image instead of added directly images. As one can see in equation 3.3, the user does not add the noise (δ) to her images directly. The variable δ can get any values from $[-\infty, \infty]$.

$$x_{i,pert} = \frac{1}{2}(\tanh(\operatorname{arctanh}(2 \times (x_i - 0.5)) + \beta \times \delta)) + 0.5 \quad (2.5)$$

where $x_{i,pert}$ is the perturbed version of the image x_i and δ is the learned perturbation.

FGS for SRCNNs: Similar to FGS attack for CNN classifiers, this method is used to learn adversarial noise for SRCNNs, but the objective function is to maximize the difference between a generated high resolution image and its original high resolution image with minimum perturbation on the low resolution image:

$$\min_{\delta} \|\delta\|_p - \mathbf{dist}(hr(I_{lr} + \delta) - I) \quad (2.6)$$

where δ , I , I_{lr} , hr and \mathbf{dist} are adversarial noise, the original high resolution image, the low resolution image, an SRCNN's function which generates the high resolution of a given image and the distance function which measures the difference between two images, respectively. Peak signal to noise ratio (PSNR), minimum square error or norm p can be used as distance function here.

Chapter 3: Adversarial Perturbation as Image Privacy Defense

3.1 Introduction

Integration of high-resolution cameras into ubiquitous handheld devices (*e.g.*, mobiles, tablets) has made it easy to capture everyday moments in digital photos. Combined with the availability of fast mobile networks many of the captured digital images and videos are shared over the network or transmitted to third-party storage providers. It has been reported that users of social networks share more than 1.8 billion new photos each day [2]. While cloud-based photo storage and sharing platforms provide virtually unlimited storage space for saving and sharing images and are highly reliable and available, such services also raise privacy concerns. Although most social networks and photo storage and sharing platforms now provide privacy settings to limit public accessibility of images, they are often complicated and insufficient [91; 108]. Furthermore, those platforms have direct access to users' images and it is not inconceivable for such service providers to exploit the users' image data for their own benefit [104; 113].

The advent of scalable Convolutional Neural Networks (CNNs) for automated face detection and recognition [61; 116; 127; 78; 56] exacerbates this concern. While such tools provide benign and beneficial service to users, they nevertheless pose a significant privacy threat. For example, Shoshitaishvili et al. [108]



Figure 3.1: From left to right: original image, UEP perturbed image, and k-RTIO perturbed image.

recently proposed a method for tracking relationships among people by analyzing pictures shared on social platforms using face recognition and detection CNNs. Recently, a face-recognition based image search application, Clearview AI app [44], has emerged that can take a given picture of a person and search a database of more than 3 billion images scraped from Facebook, YouTube and millions of other websites and provide matching images along with links to websites containing those photos. News of the use of this application by law enforcement agencies has raised concerns about severe erosion of privacy. Emergence of such applications combined with the ineffective privacy settings of social networks, necessitates the development of practical tools that allow end users to protect their privacy against automated classifiers. In this chapter, we primarily focus on thwarting automated face recognition.

Adversarial image perturbation has been recently proposed as a way to protect against automated face recognition by fooling CNNs into misclassifying the image [51; 105; 33]. Adversarial perturbations [35; 36] have the nice property that the perturbed images are perceptually indistinguishable from the original image

but are misclassified by a CNN. Thus adversarial perturbation provides a nice balance of privacy against automated classifiers without degrading the usability of the images for end users. However, these methods are not always practical for real-world use due to the unrealistic assumption of full knowledge about the adversaries' CNNs (*i.e.*, white-box model).

While black-box approaches for adversarial perturbations do exist (*e.g.*, [71; 83; 93]), they either assume that end users have access to large datasets (*e.g.*, millions of images) and can train large local CNNs (*e.g.*, with thousands of classes), or that they can query the target CNN making them unrealistic as well. Semantic adversarial perturbations [49; 48] are a newer class of adversarial perturbations that can target unknown CNNs. They do not require any learning and are therefore computationally very efficient. However, they are not reversible making them unsuited for image storage applications.

Contributions: In this chapter, we explore the practicality of adversarial perturbations to thwart automated image classification by CNNs, and make the following contributions:

1. We identify key requirements for practical adversarial perturbation based image privacy against automated face recognition.
2. We propose two novel schemes for adversarial perturbation of images that do not require special knowledge of the target CNNs (*i.e.*, black-box adversarial methods) and that satisfy the identified requirements.
 - (a) Our learning-based *Universal Ensemble Perturbation (UEP)* approach

(2nd image in Figure 3.1) learns a single transferable adversarial perturbation that can be applied to multiple images. Compared to previous adversarial perturbation learning techniques, our UEP approach requires fewer and smaller local CNNs trained over smaller (by an order of magnitude) training sets.

(b) Our *k-Randomized Transparent Image Overlays (k-RTIO)* is a novel *semantic adversarial perturbation* [48] approach that generates many unique image perturbations (3rd image in Figure 3.1) using only a small number of overlay source images and a secret key, and without requiring users to train their own CNNs. In contrast to previous semantic adversarial perturbation approaches, k-RTIO perturbations are reversible.

3. We evaluate the effectiveness of both methods against highly accurate celebrity recognition and face detection models from `clarifai.com`, `Google Vision API`, and `DeepFace` [96], two state-of-the-art online classifiers and one offline classifier, respectively. Our results show that while UEP and k-RTIO are effective at thwarting automated face recognition, k-RTIO is computationally much cheaper than UEP.
4. We discuss potential counter measures against our schemes and demonstrate one effective countermeasure against UEP.

The rest of this chapter is organized as follows: We discuss related work in Section 3.2. In Section 3.3, we present the system model and requirements for a practical privacy mechanism based on adversarial perturbations. We present

some preliminaries on adversarial examples and transferable adversarial methods in Section 3.4. We introduce our two proposed approaches in Section 3.5 and evaluate them in Section 3.6. In Section 3.7, we discuss our findings and some future directions. Finally, we conclude in Section 3.8.

3.2 Related Work

Many cryptographic and non-cryptographic techniques for protecting image privacy have been proposed. Here we primarily focus on works most relevant to the problem of thwarting recognition by automated classifiers while allowing recognition by humans.

Adversarial Perturbations: The problem of thwarting recognition by automated classifiers while allowing recognition by humans has recently received attention in the research community [51; 76; 123; 33; 105]. Adversarial perturbation based approaches [51; 33; 105] among them are most closely related to this chapter. While those approaches can fool CNNs and preserve images’ recognizability by humans they have some limitations. Oh *et al.* [51] assume that end users have full knowledge about and access to target CNNs (*i.e.*, white-box model). Other works [33; 105], while they use a black-box model, assume users are able to train large CNNs to learn perturbations. However, in real-world settings, users typically do not have access to the target CNNs or other automated tools used by adversaries or online service providers to analyze images. Moreover, all those approaches learn a perturbation for each image making it expensive to protect a large set

of images. Fawkes [105] aims to perturb images to prevent target CNNs from learning a model on them and can be complementary to our effort. Semantic adversarial perturbations [49; 48] that can fool unknown CNNs without requiring any learning have been recently proposed. As they do not require any learning, they are computationally very efficient. However, they are not reversible making them ill-suited for image storage applications. In contrast, we aim to propose practical methods for generating reversible transferable perturbations that do not need high computation or storage resources.

Image Encryption: Encryption techniques specifically designed for protecting image privacy have also been proposed (*e.g.*, [97; 126]). These techniques obfuscate the entire image and as a consequence images are rendered unrecognizable even by their owners making them unable to distinguish between images without first decrypting them. Further, it has been shown that P3 [97] is vulnerable to automated face recognition [79].

To address the problem of searching through encrypted images, Pixek App [1] uses image encryption with embedded searchable tags. However, such schemes require modifications at the service providers (this is the same for encrypted file system [34; 53] based solutions as well) and need additional effort on part of end users to tag and browse images using tags. In contrast, our focus is on techniques that are not only reversible but also allow end users to manage their image repositories naturally (*i.e.*, visually rather than using keywords/tags).

Thumbnail Preserving Encryption (TPE) schemes [128; 76; 123] have been proposed to preserve privacy of images while allowing image owners to distinguish

between ciphertexts of different images naturally (*i.e.*, visually) without having to decrypt them. They achieve this by ensuring that the ciphertext is itself an image with the same thumbnail as the original image. Depending on the size of the thumbnail preserved, TPE schemes are capable of thwarting recognition by CNNs [76; 123] while still allowing image owners to distinguish between encrypted images. Their goal was to thwart recognition by any end user other than those who are already familiar with the image, and hence preserve thumbnails with very low resolution. In contrast, our work aims to thwart only classifiers and not humans.

Irreversible Obfuscations: An early approach to thwarting facial recognition software while preserving facial details was face deidentification (*e.g.*, [87; 39; 40; 27; 52; 117]) that was proposed to provide privacy guarantees when sharing deanonymized video surveillance data. In this approach instead of obfuscating the image or parts of it, faces in the image are modified to thwart automated face recognition algorithms. These approaches extend the k -anonymity [118] privacy notion to propose the k -same family of privacy notions for images. At a high-level, to provide k -same protection for a face in an image it is replaced with a different face that is constructed as an average of k faces that are the closest to the original face. These approaches have the benefit of providing mathematically well-defined protection against face recognition algorithms. However, these approaches have the downside that these modify the original image and the transformation is typically not reversible. Further, the perturbed image is typically not correctly recognizable by humans. A slightly different but related approach is where a face that needs privacy protection in an image is replaced with a similar one from a database of 2D or 3D

faces (*e.g.*, [13; 70]). AnonymousNet [66] is recent de-identification approach using generative adversarial networks (GANs). These approaches are also irreversible. In contrast, we aim to create reversible perturbations where the perturbed images are recognizable by humans.

Approaches like blurring, pixelation, or redaction have long been used for protecting privacy in images and text (*e.g.*, [137; 60]). However, such techniques are not only *irreversible* but have been shown to be ineffective especially against automated recognition methods [137; 60; 87; 45; 79].

Approaches that remove people or objects from images (*e.g.*, [54; 11; 98]) or that abstract people or objects (*e.g.*, [22; 124]), or that replace faces [114] also exist. However, those approaches are also irreversible. Further, it has been shown that even if redaction techniques like replacing faces with black space are used, it may be possible to deanonymize using person recognition techniques [90].

3.3 System Model and Requirements

In this chapter, our focus is on practical adversarial perturbation approaches that make scalable automated image analysis by service providers more difficult. Thwarting recognition by humans is not a goal. In fact, we aim to develop approaches that allow image recognition by end users so they can interact with images naturally. We assume that image storage and sharing service providers do not have access to meta-data or auxiliary information and treat them as an honest-but-curious or semi-honest adversary [89]. We assume that the service provider

has trained an image classification CNN either using publicly available user images or images that the users have already stored with the service. A user's goal is to perturb the images that they upload to the service to prevent automated recognition or classification of their images. To be practical the perturbation mechanism should satisfy the following requirements.

Black-box Scheme: In practice, end users will not have access to the CNNs used by third-party service providers and typically will not have detailed knowledge of the target CNNs parameters and weights. Thus, any viable perturbation mechanism should not require any special knowledge of the target CNN, eliminating the use of white-box adversarial perturbation techniques. We assume that the user does not know anything about the structure of the target CNN, not even all the classes that the CNN is able to classify images into (*i.e.*, it is a black-box), and assume that the user does not have query access to the CNN.

Recognizability: While white-box (*i.e.*, full knowledge of target CNN) adversarial perturbations typically are very small and imperceptible [36], black-box perturbations on the other hand can be significant and are typically perceptible [83]. To be viable, the perturbed images should be perceptually recognizable by end users but not recognizable by automated classifiers. Users should be able to browse through perturbed images naturally using the standard interface, without having to reverse the perturbation.

Recoverability or Reversibility: For image storage applications, end users must be able to reverse the perturbation in order to recover the original image with minimum to no distortion, *i.e.*, no loss of perceptual quality. At the same time,

the service provider should not be able to remove or reduce the perturbation.

Low Computational Cost: Since end users need to create the perturbations, learning or creating perturbations should not require excessive computational effort or large volumes of training data. We assume that, while an individual user may have access to their own images and some images of other potential users of the platform (either publicly available or those of friends using the same platform), a user does not have access to the entire dataset used by the service provider for training the target CNN.

Low Storage Cost: The storage requirements of the scheme should be small and should not increase with the number of images. Any information needed to reverse the perturbation should take up minimal storage at the end user.

Compatibility: Since service providers have incentives to learn and profit from user data, proposed schemes should not depend on their support and should be compatible with existing services to ease adoption. Hence, using embedded thumbnails, encrypted file systems (*e.g.*, [34; 53]) and more advanced solutions such as searchable encryption for images [1] are not practical. Most of the popular storage services that we tested such as Google Drive, iCloud, *etc.* do not support embedded thumbnails.

3.4 Preliminaries

As discussed in Chapter 2, Convolutional Neural Networks (CNNs) are vulnerable to adversarial noise, however white-box adversarial noise are not transferable to

other CNNs. To address this issue, transferable adversarial perturbations have been proposed. In this section we introduce the well-known approaches for generating adversarial perturbations.

Ensemble Method: Ensemble methods to create transferable perturbations were proposed in [71; 8]. Those works learn a perturbation for a given image on several local CNNs, instead of using only one CNN, such that the learned perturbation fools all local CNNs. They established that such perturbations are transferable and can fool unknown CNNs with high success rate. In other words, a perturbation for an image is generalized by learning it on multiple CNNs. To show the transferability of perturbations, authors assessed their method on, clarifai.com, a state-of-the-art online classifier.

Universal Perturbation: Moosavi-Dezfooli *et al.* [83], showed that it is possible to find a single (universal) perturbation that can be applied to multiple images to successfully fool a CNN into misclassifying all of them. Universal perturbation is defined as a noise pattern δ which leads a CNN misclassifying most of the input images (x 's) with probability of p (p-value) when added to those input images. This work showed that a universal perturbation is transferable to other CNNs with different structures but trained on the same dataset as the original. Further, the perturbation is added directly to images which causes significant loss during image recovery because of rounding the pixels' value to 0 or 1 if they are not in range of $[0, 1]$.

Query-Based Black-Box Attack: An interactive attack was proposed for accessible but unknown (black-box) CNNs [93]. In this approach, the authors trained a

substitute CNN on a synthetic dataset created by sending queries to the target CNN. They revealed that the substitute CNN can approximate the target CNN’s function F very well. Nonetheless, the number of the queries required to create a CNN increase with size of the images and target CNN. Moreover, the accessibility of the target CNN to make queries is essential in this approach. Furthermore, a perturbation has to be learned for each specific image.

While all of these approaches are black-box approaches, (i) they either need to train and learn on multiple (4 – 5) large local CNN(s) (see Summary in Section 3.6.1) to create a transferable perturbation(s) [71; 8; 83] or (ii) generate one perturbation per image [71; 8] and do not meet *low computation* and *low storage* cost requirements. Further, they add the perturbation directly to image’s pixel value leading to significant losses during recovery.

3.5 Proposed Approaches

In this section, we introduce two methods to perturb images in order to fool any unknown CNN. The first method is a learning-based approach called *Universal Ensemble Perturbation (UEP)*. The second method is a semantic perturbation method called *k-Randomized Transparent Image Overlays (k-RTIO)*. We motivate each approach and describe how we generate and apply the perturbations.

3.5.1 Universal Ensemble Perturbation

As discussed in Section 3.3, any viable perturbation approach for image privacy should be black-box as users are unlikely to have access to or knowledge about CNNs used by service providers. Since black-box perturbations tend to be significant and perceptible, reversibility of perturbations becomes important to recover the original image. Storing the perturbations used for each image so the original can be recovered will require nearly the same order of storage as storing the original images themselves. Therefore, in our approach we aim to generate a *single perturbation* that can be applied to several images to successfully fool an unknown CNN, that is, both *universal* and *transferable*. We achieve this by using an *ensemble* approach through learning on a few local CNNs. Unlike previous work [83; 71], our approach uses fewer and smaller CNNs, and requires significantly smaller training datasets than the one used to train local CNNs (see Section 5.4).

Perturbation Learning: We create our *Universal Ensemble Perturbation* (UEP) approach by building on the ideas from the ensemble approach [71], universal perturbation approach [83], and CW perturbation optimization and addition approach [18].

UEP learns a highly transferable, universal, and reversible perturbation using an ensemble of a few small local CNNs. Moreover, to prevent the target CNN from reducing the impact of the perturbation by using a reduced resolution version of the image [26], we learn a perturbation which works for different image resolutions by training each local CNN on different resolutions of the image. For this work, we

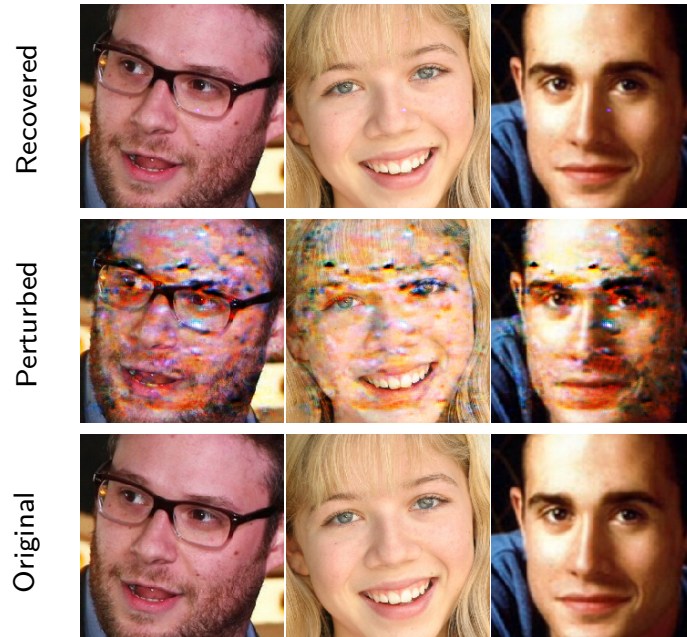


Figure 3.2: From bottom to top: The last row shows the original images of celebrities. The middle row represents the UEP perturbed images. The first row displays the recovered images.

used three small CNNs with only 10 classes each (see Section 3.6). We empirically found that using fewer than three CNNs makes learning transferable perturbations harder, while using more than three CNNs makes learning a perturbation computationally expensive without significant improvement in success rate.

After learning a few (in our case only 3) local CNNs on different image resolutions, we learn a universal perturbation which can fool all the local CNNs with a given probability of p ($\in [0, 1]$). To optimize the misclassification rate, we extend the objective function introduced in [18] for an ensemble of CNNs and a universal

```

UEP( $Y_1(x), \dots, Y_m(x), x_1, \dots, x_n, p$ ):
 $\delta = inf$ 
// finding the true label of each image
 $\forall i \ y_i^* = argmax(Y(x_i))$ 
 $lbound = 0.001, upbound = 100000$ 
while  $lbound < upbound$ :
   $c = (upbound + lbound)/2$ 
   $\forall i, z_i = x_i$ 
  while  $\sum_i \prod_k \sigma(Y_k(z_i) \neq y_i^*) < p \times N$ 
     $\delta' \leftarrow \min_{\delta} \|\delta\|_2 + cf(z_1, \dots, z_N)$ 
     $z_i = \frac{1}{2}(\tanh(\arctanh(2 \times (x_i - 0.5)) + \delta')) + 0.5$ 
  if  $\|\delta'\|_2 < \|\delta\|_2$ :
     $\delta \leftarrow \delta'$ 
     $upbound = (upbound + lbound)/2$ 
  else
     $lbound = (upbound + lbound)/2$ 
return  $\delta$ 

```

Figure 3.3: UEP method learns a universal perturbation δ on a few local CNNs (Y_1, \dots, Y_c) for set of given inputs x_1, \dots, x_n .

attack as follows:

$$f(x_1, \dots, x_N) = \sum_l \sum_i \max(\max\{Y_l(x_i)_j : j \neq y_i^*\} - Y_l(x_i)_{y_i^*}, -\kappa) \quad (3.1)$$

where $Y_l(x_i)$ is the probability vector assigned by the l^{th} CNN to the i^{th} input which shows the probability of the input image belonging to each category. And y_i^* denotes the true label of x_i . We aim to minimize the perturbation amount as well as maximize the misclassification confidence (κ). The problem can be defined

as follows:

$$\min_{\delta} \|\delta\|_2 + cf(x_{1,perturbed}, \dots, x_{N,perturbed}) \quad (3.2)$$

A large value for coefficient c causes the resulting perturbation to have a larger value (to force all CNNs to misclassify it), while a very small value for this parameter may lead to null solution space since it tries to fool all CNNs for all inputs with a small perturbation. We optimize the perturbation for different coefficient c values using a binary-search (see Figure 3.3) and pick the smallest perturbation that can fool $p \times N$ of total samples (N).

Perturbation Addition: In the universal perturbation approach [83], the perturbation vector is added directly to all images, and if the new value of pixels are not in the range of $[0, 1]$, then the new values are rounded to the nearest acceptable values (0 or 1). However, rounding a pixel’s value leads to information loss during image recovery. Therefore, we add the learned perturbation to the arc-tangent hyperbolic value of an image instead of adding directly to the images as in [18]:

$$x_{i,pert} = \frac{1}{2}(\tanh(\operatorname{arctanh}(2 \times (x_i - 0.5)) + \beta \times \delta)) + 0.5 \quad (3.3)$$

where $x_{i,pert}$ is the perturbed version of the image x_i and δ is the learned perturbation. Here β is a weighting factor that tunes transferability versus perturbation amount. While this does not eliminate rounding losses completely, it significantly reduces such loss.

Further, unlike the traditional adversarial approaches, UEP perturbation can

be added in with different weights to the image to control the trade-off between transferability and recognizability. While learning UEP on local CNNs, we set $\beta = 1$ to create a powerful perturbation. But, when applying the learned perturbation to an image, we can set β value to trade-off between fooling success rate and recognizability (see Section 3.6). The original image is recovered by reversing the process on the perturbed image. Adding the perturbation to the arc-tangent hyperbolic value of an image, both simplifies the objective function and alleviates information loss during image recovery [18].

Practicality of UEP: As the preceding discussion shows, UEP is designed to be a black-box perturbation approach that produces highly transferable and universal perturbations. It meets the reversibility and low storage cost requirements identified in Section 3.3 as perturbations can be reversed by storing only the universal perturbation. While the perturbations produced by UEP are significant and perceptible, as seen in Figure 3.2, the perturbed images are still recognizable for humans and can be recovered with very low loss by removing perturbation. Further, since UEP uses three small local CNNs with only 10 classes and does not require users to have access to large datasets. UEP is also computationally more efficient compared to previous approaches as will see in Section 5.4. Further, UEP does not require any changes to the service provider and can work with existing services.

3.5.2 k-Randomized Transparent Image Overlays

While our UEP approach requires lower computational resources compared to previous learning-based approaches, learning perturbations is still computationally expensive (see Section 3.6.1). Semantic perturbations [49] on the other hand do not rely on learning and tend to be computationally cheaper. Different semantic perturbation approaches target different weaknesses of CNNs (*e.g.*, coverage of training data, reliance on inter-pixel dependencies etc.) [48].

The main component of CNNs are kernels in the convolutional layers. Kernels are small windows moving over image space and measuring dependencies among adjacent pixels to find a pattern or a feature. Transparent image overlay-based

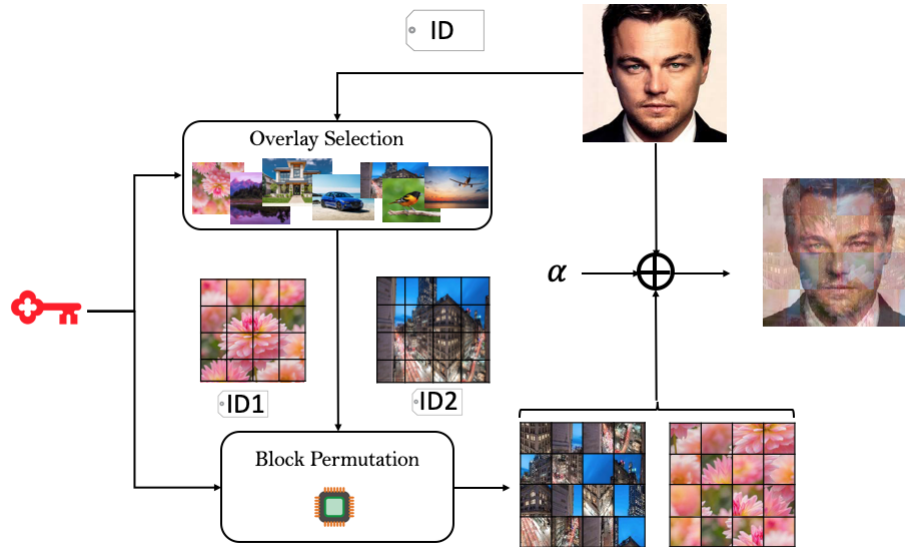


Figure 3.4: k-RTIO Scheme. ID of main image is used to select k overlay images from the set S and then ID of selected images are used to generate a permutation. Both overlay selection and block permutation algorithm use the user’s secret key.

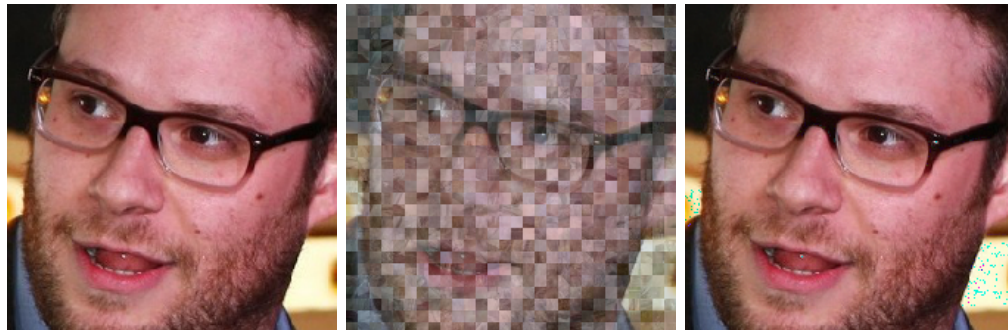


Figure 3.5: Left to right: original, k-RTIO perturbed and recovered images.

approach tends to decrease the pixel dependencies and makes features invisible to CNN kernels. Here we build on this transparent overlays based approach that was originally proposed for fooling Google Video API [49]. When translated to images, this approach adds a weighted overlay image to the original image as follows:

$$I_{pert} = \alpha \times I_{org} + (1 - \alpha) \times I_{ovl} \quad (3.4)$$

where α is mixing parameter, and I_{pert} , I_{org} and I_{ovl} are perturbed, original, and overlay images.

Despite the high transferability (success in fooling unknown CNNs) especially at low values of α , direct application of this technique has the following drawback. If the same overlay image is used for perturbing multiple original images then the original images can potentially be recovered through a well known signal processing technique called Blind Signal Separation [15] (see Figure B.2 in Appendix B.2). If a different overlay image is used for each original image, then all those overlay images need to be stored locally to be able to recover the original images defeating

the purpose of cloud-storage and violating our low storage requirement. Further, we need to make sure that perturbed image is recognizable by end users. To overcome this problem, we propose *k-Randomized Transparent Image Overlays* which generates a unique overlay for every image using only a small set of initial overlay images, a secret seed and a pseudorandom function.

Perturbation Generation: To generate *unique* overlays for each original image, we first choose a random subset of k (typically 3 or 4) candidate overlay images from a set S of stored initial overlay images. This set S is relatively small (30 images) and can be stored encrypted on a cloud platform or locally. The choice of this set does not significantly change the fooling performance of k-RTIO; we discuss selection of this set in Section 3.6.2. As shown in Figure 3.4, to select these k images for each source image, users use their secret key and ID of the original image (each image has a different ID) as inputs to a pseudorandom function. More concretely, we employ AES as the pseudorandom function. Let key denote a 128-bit seed and let id be a unique identifier for the given source image (e.g., a filename or timestamp). Then the choice of the j th overlay image (out of $|S| = m$ possibilities) is made according to:

$$\text{AES}(key, id||0||j) \bmod m.$$

After selecting k random candidate images, the k chosen images are divided into b blocks each and these blocks are permuted using a pseudorandom permutation. The permutation of blocks is based on the candidate overlay image ID, source image ID, and user’s secret key. Specifically, the blocks of the j th overlay image

are permuted using the Fisher-Yates shuffling algorithm [32]. Recall that in the i th iteration of Fisher-Yates, the algorithm chooses a random value in the range $\{0, \dots, b - i\}$, where b is the number of items being shuffled. This random choice can be derived in a repeatable way (for the j th overlay image) via:

$$\text{AES}(\textit{key}, \textit{id} \| 1 \| j \| i) \bmod b - i + 1.$$

As long as the *ids* of a user’s images are distinct, then all inputs to AES are distinct. Use of a pseudorandom function guarantees that all its outputs are indistinguishable from random, when invoked on distinct inputs. Using this approach, the user needs to store only the short AES *key*, and the set of initial images S from which all unique overlay images can be re-derived as needed for recovering the original image (see Figure. 3.7).

The final overlay image is simply the average of these k images with permuted blocks. Although the pool S of base overlay images may be small, the number of derived overlay images that result from this process is very large. For example, for an image set of $|S| = 10$, $k = 4$ and $b = 12$, the number of possible overlay images is $\frac{10!}{10!(10-4)!} \times (12!)^4 \approx 7.3 \times 10^{31}$ making it hard for the target CNN to correctly guess the overlay used even if S is known.

Perturbation Addition: As shown in Figure 3.4, the generated overlay is added to the source image as follows:

$$I_{\textit{pert}} = \alpha \times I_{\textit{org}} + \frac{(1 - \alpha)}{k} \sum_{I_{\textit{ovl}_i} \in \textit{Overlays}(\textit{key}, \textit{id}, b, S)} I_{\textit{ovl}_i}$$

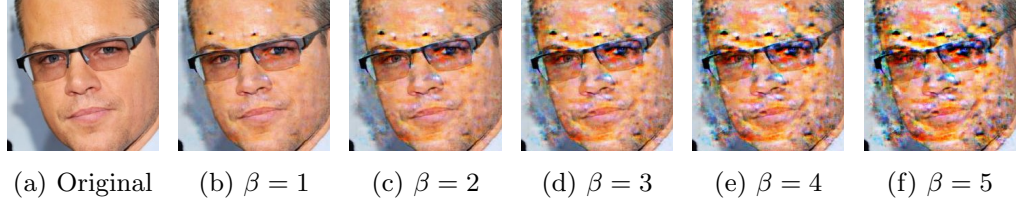


Figure 3.6: UEP recognizability vs. β that controls noise level

where $\text{Overlays}(key, id, b, S)$ is a function that takes secret key, image id, number of blocks and overlays set S and returns k overlays images with permuted blocks. The perturbation can be removed to recover the original image using the reverse process as follows:

$$I_{org} = \frac{I_{per}}{\alpha} - \frac{(1 - \alpha)}{k \times \alpha} \sum_{I_{ovl_i} \in \text{Overlays}(key, id, b, S)} I_{ovl_i}$$

Practicality of k-RTIO: As the preceding discussion shows, k-RTIO does not require knowledge of the target CNN, nor does it require support from the service provider. Further, k-RTIO generates semantic perturbations and does not require learning using any CNNs and is therefore computationally cheaper than UEP (see Section 3.6.2). Since the perturbations are efficient to generate, end users do not need to store individual perturbations for recovering original images but can recreate them using a secret key and a small set of initial overlay images leading to low near constant storage overhead. The recognizability of images perturbed using k-RTIO is a function of α , b and k . In Section 3.6.2 we will explore values of these parameters such that k-RTIO can thwart recognition by automated classifiers while providing reasonable recognizability for humans.

3.6 Evaluation

In this section, we evaluate the success rate of the two proposed methods against two state-of-the-art online image classifiers (www.clarifai.com and Google Vision API) and a state-of-the-art offline recognition and detection model DeepFace [96]. We then discuss the computational costs of each approach. Finally, we explore different filtering techniques that classifiers could potentially use as countermeasures against the proposed perturbations.

```

Overlays(key, id, b, S):
   $K = \emptyset$ 
  for  $j = 1$  to  $k$ :
    // randomly choose  $k$  overlay candidates
     $v = \text{AES}(\textit{key}, \textit{id}||0||j) \bmod |S|$ 
    IMG =  $v$ th element of  $S$ 
    remove  $v$ th element from  $S$ 

    // keyed Fisher-Yates shuffle on blocks
    for  $i = 1$  to  $b$ :
       $t = \text{AES}(\textit{key}, \textit{id}||1||j||i) \bmod b - i + 1$ 
      swap blocks  $b - i$  and  $t$  in IMG

    add IMG to  $K$ 

  return  $K$ 

```

Figure 3.7: Pseudorandom method for creating overlay images with permuted blocks for an image with identifier id . # of blocks per overlay image is b . Set of candidate overlay images is S .

3.6.1 UEP Performance

UEP Simulation Setup: *UEP* requires training a few small local CNNs. We trained 3 small CNNs, two VGG-16 [111] and one Facescrub CNN [79] using random initialization points. VGG-16 has 5 convolutional layers and 2 fully connected layers (see Figure 2.1) and FaceScrub CNN has 3 convolutional layers and 2 fully connected layers (more details of the CNNs are in Appendix A.1). Recall that, to prevent the target CNN from reducing the impact of the perturbation by using a lower resolution version of the image [26], we train each local CNN on different image resolutions and learn a perturbation over them. To implement a resolution reduction or a thumbnail function, we used an additional convolutional layer size of 2×2 with values of $\frac{1}{2 \times 2} = 0.25$ to resize input images before rendering them to CNNs. For example, as shown in Figure 3.8, by setting up the stride value to 2×2 we can have a thumbnail function that reduces the size of the image to $\frac{1}{2 \times 2} = 0.25$ of original size .

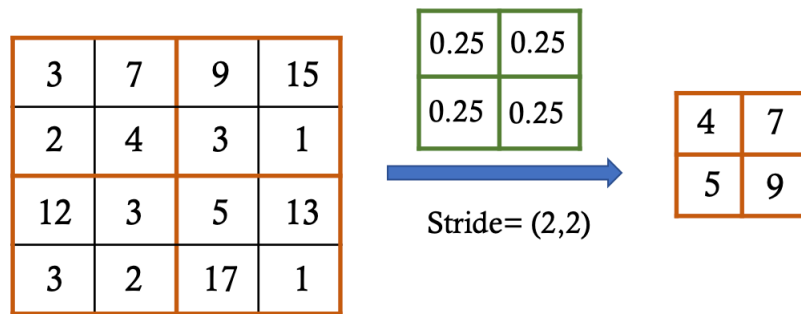


Figure 3.8: Using a convolutional layer with stride value of (2,2) and kernel values of 0.25, one can implement a thumbnail function to reduce the size of an image to 25% of original size.

We used the FaceScrub [88] dataset, an image dataset of 530 celebrities (classes) and $\approx 100K$ images for our evaluations. We selected 10 classes, uniformly at random, from among 230 classes of FaceScrub that have at least 150 images in the training dataset. The use of only 10 classes for training local CNNs is keeping inline with our assumption of limited end user access to data and computational capacity. To detect celebrity faces and crop them, we used DeepFace face detector CNN [96]. For each of the 10 celebrities (classes), we randomly selected 150 – 200 images from the training dataset and learned three local CNNs, two at 224×224 pixel resolution and one at 112×122 pixel resolution. We then chose 200 (N) images *in total* over these 10 celebrities (classes) from the training dataset and learned a universal ensemble (UEP) perturbation over these images across the three local CNNs that we trained. As shown in Figure 3.3, to find the best value for c parameter (see Equation 3.2), which controls the perturbation amount, we used binary search in the range of $[0.001, 100000]$ and set $p = 0.7$.

UEP Effectiveness: We evaluate UEP against Google Vision API’s face detection model, and face detection and recognition models of clarifai.com and DeepFace [96]. To this end, we selected more than 1000 FaceScrub images in total from all the 530 classes (not just the 10 classes on which local CNNs were trained). Then we cropped (Note: all adversarial perturbation methods learn on a specific image size that depends on the CNNs’ input size.) faces with DeepFace face detector model [88] and then applied UEP generated perturbation. Moreover, we can control the transferability of UEP perturbation by using different weighting (β) parameter (see equation 3.3). We evaluate our method for different values of

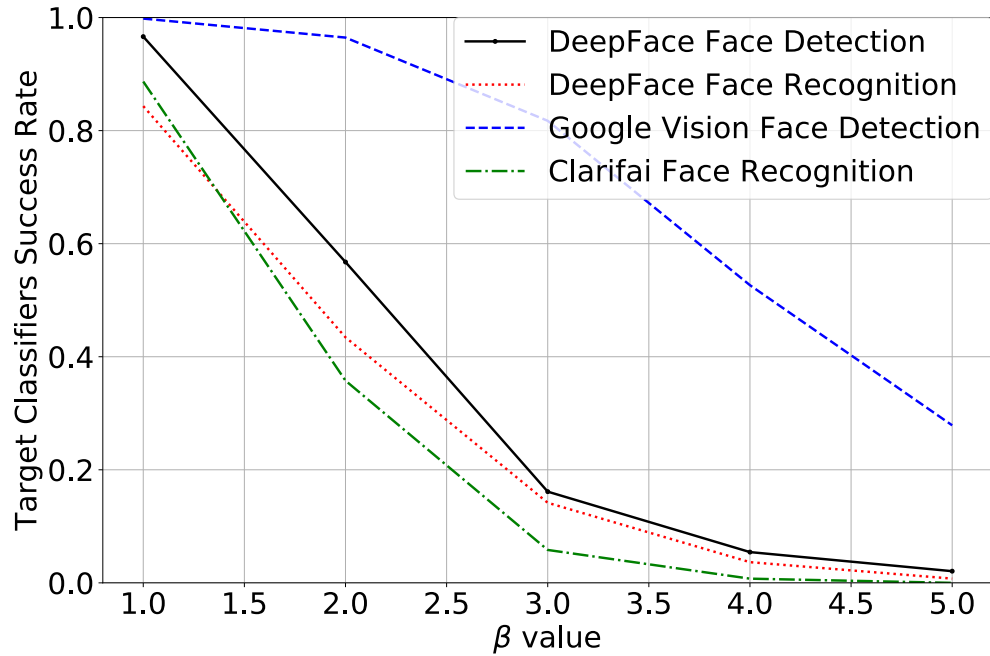


Figure 3.9: UEP performance on Google Vision API face detection model, and DeepFace face detector and recognition models.

β parameter.

UEP vs. Google Vision API: Google Vision API was able to detect 98.8% of faces in cropped benign images. As shown in Figure 3.9 (dashed blue line), for a small value of β , UEP perturbation exhibited low transferability (*i.e.*, success rate) against Google Vision API. However, for larger β values, Google Vision API’s face detection rate decreases significantly ($\leq 30\%$ at $\beta = 5$). While larger β values can fool face detection models, they also decrease recognizability for humans. Figure 3.6 shows the recognizability of the images for different values of β . Thus, there is a trade-off between fooling success rate and human recognizability. Note that, this is a face detection model and not a recognition model.

UEP vs. Clarifai.com: Clarifai.com’s face detection could detect more than 99% of faces and recognize 87.25% faces of unmodified images. Clarifai.com was able to detect 98% of perturbed faces with UEP perturbation even with $\beta = 5$. However, its face recognition was able to recognize fewer than 6% of faces for $\beta = 3$ (green dash-and-dot line in Figure 3.9). In other words, face recognition model is more sensitive to perturbations. While small values of β are not good enough to thwart recognition by unknown CNNs, with $\beta = 2$, face recognition model of clarifai.com was able to recognize only 37% of faces and this comes down to less than 6% for $\beta = 3$. As seen in Figure 3.6, perturbed faces still remain recognizable for $\beta = 3$.

UEP vs. DeepFace Models: We also evaluated UEP against DeepFace face detection and recognition models. We sampled more than 1000 images of celebrities from FaceScrub dataset that are known to DeepFace face recognition model (about 186 classes). This model was able to detect all faces in unperturbed images and recognize 97.28% of them. We then tested DeepFace with perturbed images with different levels of perturbation (β values). As shown in Figure 3.9, the face detection and face recognition models have low accuracy, less than 20% (for both detection and recognition) for larger values of $\beta (\geq 3)$.

Recoverability/Reversibility: UEP does not add the perturbation directly to pixels but adds it in the arc-tangent hyperbolic space (see equation 3.3). When mapping back to the pixel space (integers values in $[0, 255]$), we may need to round the pixel’s value. However, this loss is not perceptually recognizable as measured by the structural similarity index (SSIM) [138] between the original and recovered images. SSIM is a popular measure used to compare perceptual image similarity.

For two identical images SSIM value will be 1. Our results show that the average SSIM measured over the 1000 original and recovered images is 0.99 ($SD = 0.0038$). A distribution of SSIM values for UEP recovered images is shown in Figure 3.14.

UEP Computational Cost: Our universal transferable perturbation is learned using only 200 randomly selected images but can be applied to any other image. Compared to previous approaches [71; 83], our hybrid approach requires significantly less computational effort and time and achieves better results. But even training small CNNs is computationally expensive operation. For instance, training three small local CNNs and learning a universal transferable perturbation using 200 images took 13 hours on a standard desktop with 64GB RAM and an Intel Xeon E5-1630v4 3.70GHz quad-core CPU, and a GTX 1080 Ti GPU with 11GB RAM (Appendix A.1 has network details). Training the 3 local CNNs took less than 1 hour and learning the universal transferable perturbation over 200 images took about 12 hours. However, this is a one-time computation cost that can be expended offline. The learned perturbation can be used for perturbing multiple images and hence the learning cost is amortized over all the images.

Since a single perturbation can be applied for a batch of images, the cost for storing the learned perturbations in order to recover the original image is only a small fraction of the storage required for the images. For example, in our evaluation we created a universal perturbation for 200 images and hence the additional cost for local storage is one perturbation ($1/200th$ of total image storage).

Summary: We showed that even with access to a small dataset, 10 classes and at most 200 images per class, one can learn small CNNs and generate a universal

perturbation with high transferability. This is in contrast with Liu *et al.* [71] who train 4 large CNNs (with 1000 classes each) using a training dataset (ILSVRC [102]) of $1.2M$ images to achieve only 76% transferability (over 200 test images) on Clarifai.com. Similarly, Moosavi-Dezfooli *et al.* achieved only 70% fooling rate even after using 4000 images to learn a universal perturbation. When using only 1000 images to learn a perturbation they achieved less than 40% fooling rate (see Figure 6 in [83]). In contrast, UEP can learn an effective universal perturbation using only 200 images and achieve $\geq 94\%$ fooling rate.

Our evaluation of UEP, on more than 1000 images and 3 different face detection and 2 different face recognition models, shows that fooling face detection models is harder than face recognition models. In other words, face recognition models are more sensitive to perturbations. Also, a $\beta = 3$ showed sufficient fooling capability on face recognition models while keeping the images recognizable. Our UEP method also has less loss compared to other transferable perturbation generation approaches because perturbation is added in the arc-tangent hyperbolic space of the images instead of directly to their pixel values.

3.6.2 k-RTIO Performance

k-RTIO Simulation Setup: Recall that k-RTIO requires a set S of initial overlay images. We observed empirically that the choice of this set of images did not significantly alter the fooling performance of k-RTIO. However, we felt that including images with human faces may reduce recognizability. So we chose a set

of 30 random images from the Internet that did not include faces from our initial set S (this set is shown in Figure B.1 in Appendix B.1). Similar to UEP evaluation, we selected 1000 images from the FaceScrub dataset uniformly at random and applied k-RTIO perturbations that were generated using different values for α (mixing parameter), k (number of overlay source images), and b (number of blocks in an overlay image).

Effectiveness of k-RTIO: Similar to UEP evaluation, to assess k-RTIO, we evaluated it against Google Vision API’s face detection model, online face detection and recognition models of Clarifai.com, and face detection and recognition models of DeepFace.

k-RTIO vs. Google Vision API: Google Vision API detected faces in less than 40% of original unperturbed images (here we did not crop the faces but rather submitted the original images). Then we perturbed those images for which Google Vision API correctly detected faces with k-RTIO approach. Note that unlike UEP that needs to resize images to apply perturbation, k-RTIO does not need to resize the main image (just overlay images should be resized to main image’s size). As shown in Figure 3.10, Google Vision API only can detect faces in less than 35% of images when $k = 3$ and $\alpha \leq 0.5$ for small block sizes (less than 10). Increasing α value or number of overlay images (k) leads face detection success rate to increase. This is because a larger α value implies more of the original image is present in the perturbed image and thus can increase recognizability for both humans and automated classifiers. This can be seen in Figure 3.12 where the image becomes clearer for higher α values for every block size. Increasing the number of overlay images

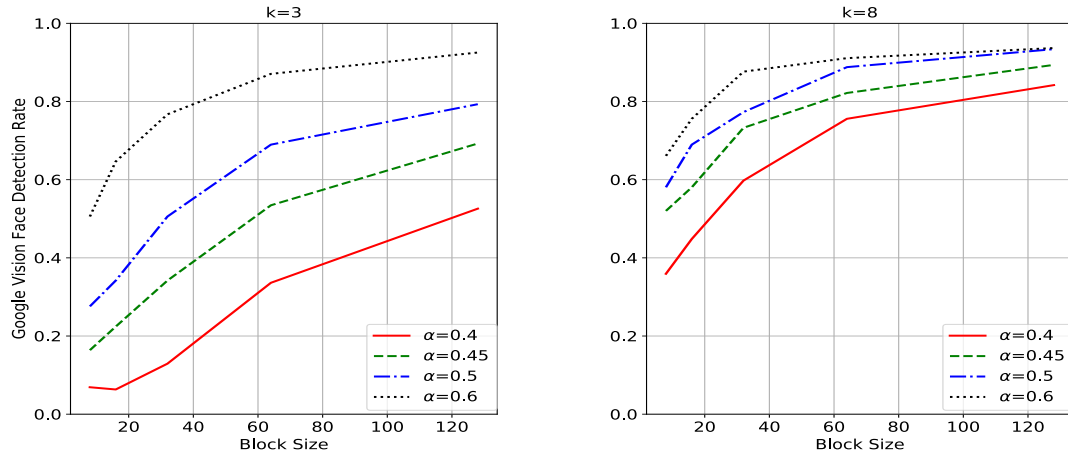


Figure 3.10: Google Vision API face detection model performance on kRTIO images for $k=3$ and $k=8$.

used (k) leads k-RTIO perturbation to look like random noise as the different overlay images are averaged, and therefore the perturbation becomes ineffective. This is evident from Figure 3.13 where for k values greater than 4 images become easy to perceptually recognize for all block sizes. Interestingly, both Figures 3.12 and 3.13 show that smaller block sizes seem to produce less obfuscated images for humans while still thwarting automated classifiers.

k-RTIO vs. Clarifai.com: Clarifai.com’s face detection model was able to detect faces in all original images sampled from the FaceScrub dataset. Clarifai.com’s face recognition model was able to recognize $\approx 82\%$ of the celebrities in these images. To evaluate the k-RTIO performance on face recognition model, we used only the images that were recognized by Clarifai.com’s face recognition model. As shown in the top row of Figure 3.11, similar to Google Vision API, for small values of $\alpha \leq 0.45$ and small block sizes clarifai.com’s face recognition model is able to

detect faces in very few images ($< 7\%$) even when face detection model is able to detect faces in more than 80% of images successfully.

k-RTIO vs. DeepFace Models: We also evaluated k-RTIO against DeepFace face detection and recognition models. As in the case of UEP, we selected more than 1000 images from FaceScrub dataset that DeepFace classified correctly. We perturbed the selected images using k-RTIO with different α values and block sizes and tested them against DeepFace. As shown in the bottom row of Figure 3.11, both models showed low performance for k-RTIO perturbed images with a small block size (even for larger α values).

Recoverability/Reversibility: We can always regenerate the exact overlay image that was added to original images using the secret key. However, k-RTIO still needs to round up the pixel values after adding overlay images leading to some loss. However, this loss is not perceptually recognizable. To measure the perceptual similarity, we again used SSIM to evaluate the similarity between the original and recovered images. Our results show that SSIM measure between original and recovered images averaged over the 1000 tested images is 0.96 (SD=0.07). A distribution of SSIM values k-RTIO recovered images is shown in Figure 3.14. k-RTIO recovered images tend to exhibit lower SSIM values as the perturbation is applied to the whole image and not just the faces as is done for UEP. For this experiment, we set the k-RTIO parameters as $k = 3$, block-size= 8 and $\alpha = 0.45$ (a sweet spot for fooling unknown face recognition models).

Computational Costs of k-RTIO: Unlike transferable perturbation generation, the *k-RTIO* method is computationally much more efficient. While it does involve

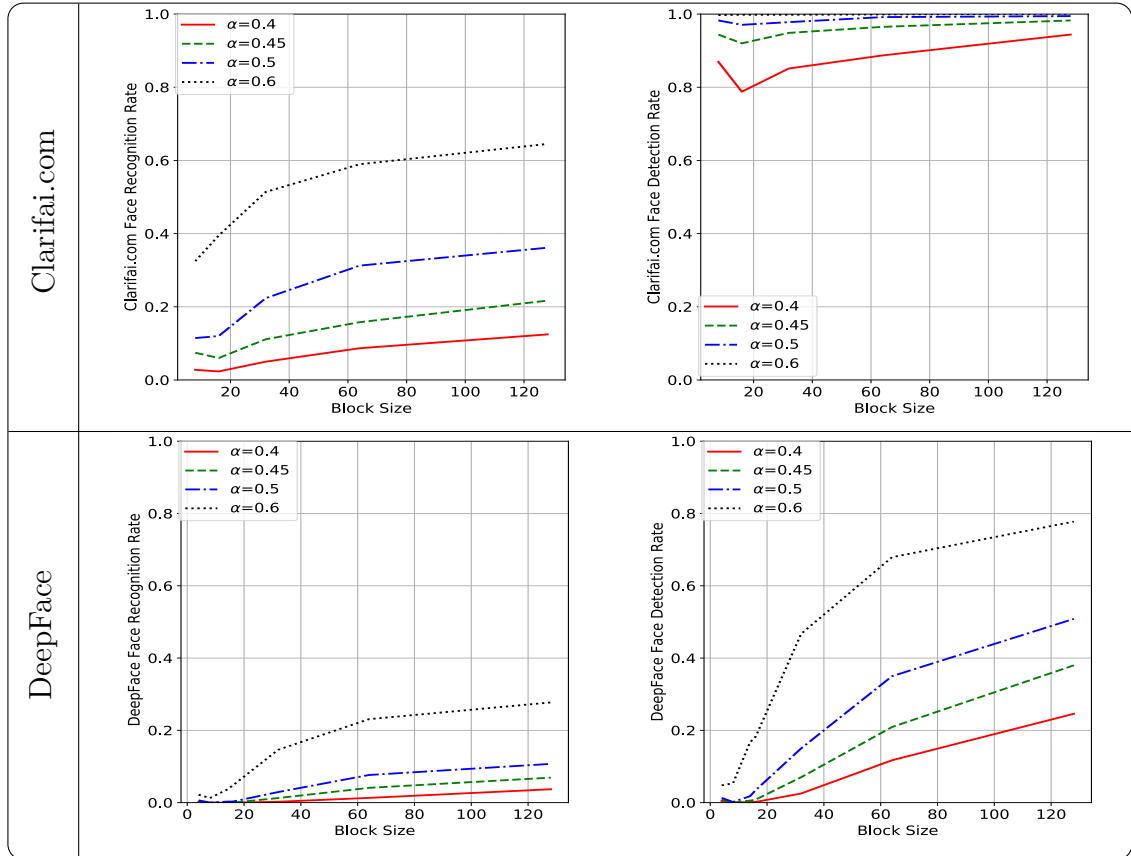


Figure 3.11: Clarifai.com (online) and DeepFace (offline) face recognition and detection models on k-RTIO perturbed images for $k = 3$.

$(k \times b) + k$ AES invocations for generating a unique overlay for each image, AES is computationally inexpensive especially with hardware instruction set support. For example, generating a permutation for 500 blocks takes 9×10^{-5} milliseconds and permuting an overlay image with 625 blocks of 40×40 pixels each took less 5×10^{-4} milliseconds (wall-clock time) with no special optimizations or effort. Further, creation of unique overlays can easily be parallelized when perturbing multiple images. End users need only store the secret key (128 bits) and a small

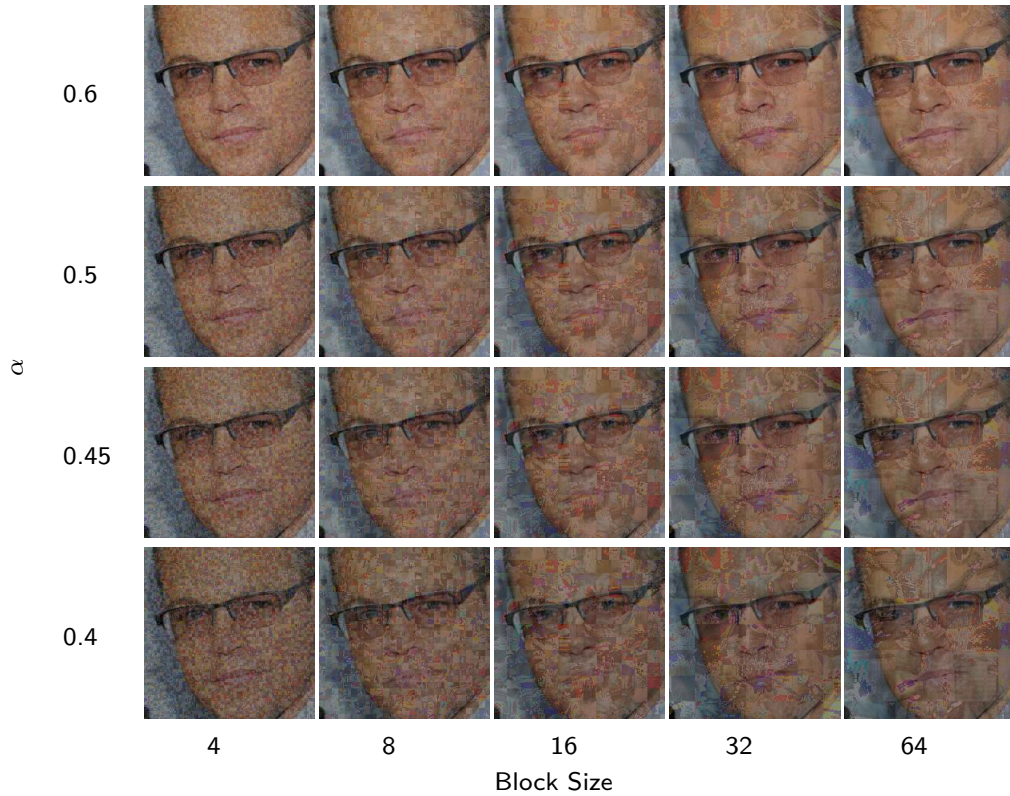


Figure 3.12: Different α values vs. size of block for $k = 3$ overlay images. Overlay images were chosen randomly but the same overlay images were used in all cases.

image set S used in the creation of unique overlays to recover the original images. Thus, storage costs remain constant irrespective of the number of images that are perturbed.

Summary: Similar to the case of UEP, we find that fooling face detection models is harder than fooling face recognition models. As shown in Figures 3.10 and 3.11, face detection models can detect faces better for larger block sizes (*i.e.*, smaller number of blocks (b)) even at lower α values. However, k-RTIO is able to effectively thwart face recognition models. For small block sizes (4×4 or 8×8), face recognition

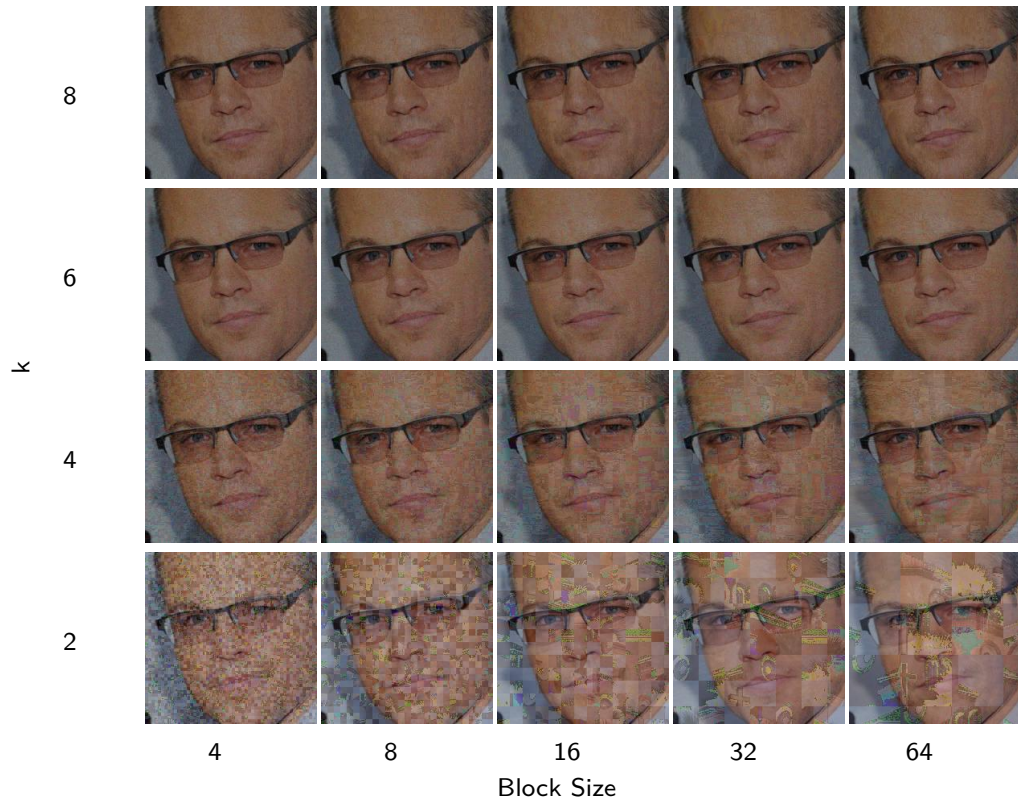


Figure 3.13: Different number of overlay images vs. different number of block sizes for $\alpha = 0.6$. Used same overlay images per row.

models recognize 15% of faces at best for $\alpha \leq 0.5$. For lower alpha values, say $\alpha = 0.4$, face recognition rate does not go above 15% for any block size. Thus, the mixing parameter α and the number of blocks b are critical factors that impact face recognition. Similarly, as shown in Figures 3.13 and 3.10, number of overlay images used is also a critical factor as using more overlays leads to a final perturbation that looks like random noise and therefore ineffective.

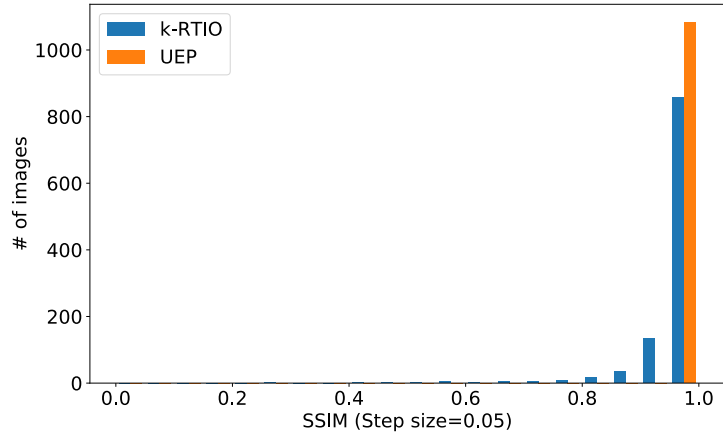


Figure 3.14: SSIM histogram for recovered UEP and k-RTIO images. In contrast to k-RTIO, UEP only perturbs faces in an image, therefore it has larger SSIM on average.

3.6.3 Potential Attacks Against UEP and k-RTIO

We explore different countermeasures that can potentially be deployed against the proposed perturbation approaches. We show that using one perturbation for several images provides adversaries with an opportunity to estimate the perturbation and recover classifiable images (if not original images). This is a downside of using a single semantic perturbation for multiple images and led to the k-RTIO approach which uses a unique perturbation per image. We also explore a countermeasure where the target CNNs learns on perturbed images.

Perturbation Estimation and Removal: Let us assume that the perturbation is added directly to images ($x'_i = x_i + \delta$) and an adversary wants to find δ given only the perturbed images. Adversaries can treat δ as the main signal and the images as noise added to the main signal. In this case, the noise signal is not independent or zero mean (but rather images of faces). However, target CNNs

do not need to recover the exact perturbation as they do not have to recover the exact original image. They only need to recover enough of a likeness to the original image to be able to classify the image correctly. To get a rough estimate of the perturbation, a target CNNs can compute the median of pixels in all perturbed images to approximate the δ . It can then add the inverted median to a perturbed image as a new layer ($x = \lambda \times x'_i + (1 - \lambda) \times inv(\delta')$).

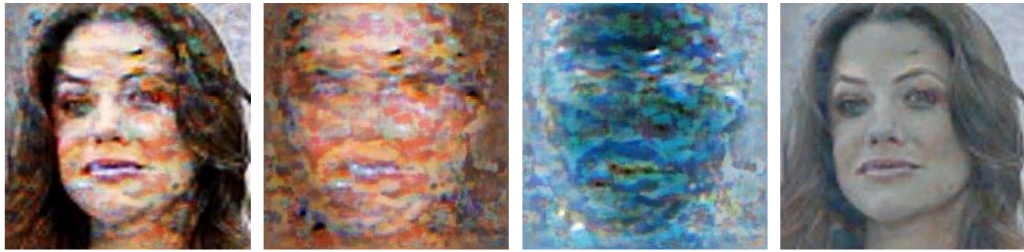


Figure 3.15: UEP perturbation estimation and removal. The first image is the perturbed image, the second one is median of 200 perturbed images. The third image is inverted median and the final image is the recovered image by an adversary ($\lambda = 0.42$).

As shown in Figure 3.15, using this approach a target CNN can obtain an image close to the original and that seems to be sufficient for recognition. We have assessed 200 images after removing the perturbation with median estimation method and submitted it to the `clarifai.com`'s celebrity face detection model. As shown in Figure 3.16, `clarifai.com`'s success rate for different values of λ (0.40 to 0.56) is as high as 70% (for $\lambda \in [0.44, 0.48]$) - more than tenfold improvement! In the worst case scenario, if we assume that the target CNN can know when it classified correctly then it can vary λ for every image and reach an accuracy as high as 80.5% with access to only 35 images modified using the same perturbation. Thus, using a single UEP perturbation for several images may allow target classifiers to estimate

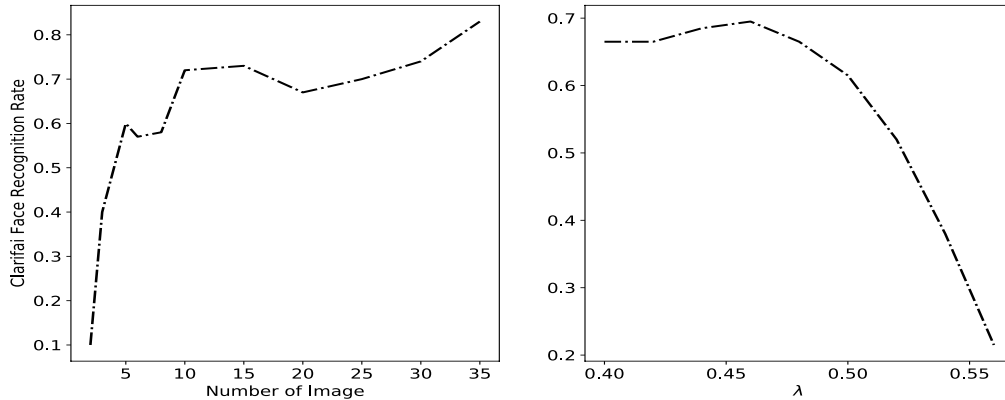


Figure 3.16: Target CNN can recover 60% of the images by using Perturbation Estimation and Removal method just with 5 images using the same perturbation. By setting λ value to 0.48, an adversary can improve its success rate to 70%.

the perturbation and recover classifiable images. As shown in the Figure 3.16, an automated classifier can improve their classification accuracy to 70% by estimating perturbation using as few as 10 perturbed images.

Unlike UEP that uses a single perturbation for several images, k-RTIO generates a unique perturbation for each image, hence this perturbation estimation and removal approach is not effective against k-RTIO.

Robust CNNs: Many methods have been proposed to detect and reject adversarial examples (*e.g.*, [80; 131; 112; 4; 6]). These methods are suitable for safety sensitive application such as disease recognition and autonomous cars in which making wrong decision may cause huge consequences. However, such approaches are not useful as defenses against UEP or k-RTIO as those methods want to detect and reject perturbed images as much as possible which aligns with our goal of preventing automated classifiers from correct classification. Other proposed methods

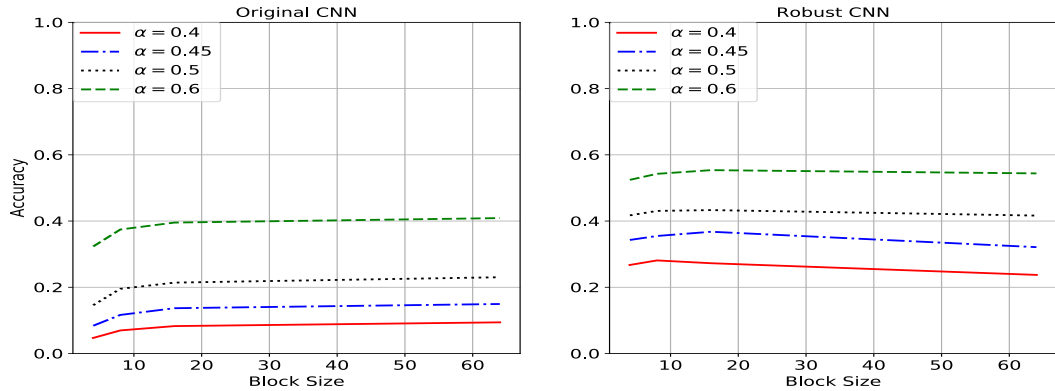


Figure 3.17: The accuracy of the CNN of TinyImageNet on perturbed images by k-RTIO images for $k = 3$ (left) for the original CNN learned on benign images, and (right) robust CNN which is learned on the k-RTIO images.

tend to make learning white-box attacks harder for an adversary [7; 93]. However, such defenses are still vulnerable to black-box attacks in which an adversary learns highly transferable perturbations on using local CNNs. Another approach is to train robust CNNs that are able to classify adversarial images correctly [74] by re-training the network on both benign and adversarial images. However, it has been shown that this approach does not make CNNs robust against all kinds of adversarial examples even for small CNNs [106]. Here we aim to assess the *classification accuracy improvement* on k-RTIO perturbed images when one trains a CNN both on benign and k-RTIO perturbed images.

For this, we selected TinyImageNet [62] with 200 classes, 100000 training-set images, and 10000 test-set images as the target CNN. We trained an Xception Convolutional Neural Network [24] on this dataset. In 100 training epochs, this CNN achieved 65.42% accuracy on the test-set images. We then added 100000 and 10000 k-RTIO perturbed images to training and test respectively, and retrained the

Xception CNN. We used the same learning rate and same random seed number to initialize the CNN weights. The robust Xception CNN achieved 68.50% accuracy on benign test-set images and 32.1% accuracy on average on k-RTIO images ($\alpha \in \{0.45, 0.5\}$, $block - size \in \{4, 8, 16, 64\}$).

As shown in Figure 3.17, learning on k-RTIO images can improve accuracy. However, the accuracy on images with low alpha values $\alpha = \{0.4, 0.45\}$ was still less than 29% and 39% respectively. In other words, robust CNNs can improve their accuracy on k-RTIO perturbed images but not significantly for small α values. Note that we used the same overlay set for perturbing images in the training and test sets to evaluate the robust CNN. This simulates a situation when the target CNN has access to a user’s overlay set S .

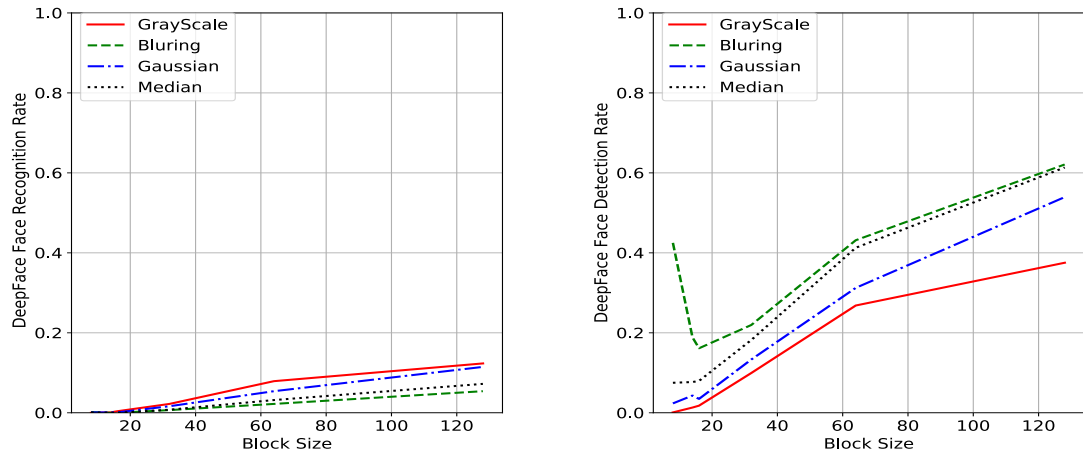


Figure 3.18: Filtering results on DeepFace face detection and celebrity recognition models for $\alpha = 0.45$ and $k = 3$.

Noise Reduction Techniques: Adversarial training showed good performance against adversarial examples for small CNNs, but this approach has been shown to be com-

putationally expensive at large scale [7] and not a desirable solution for dealing with adversarial samples.

Approaches to recover classifiable images from perturbed ones using different filtering techniques have been explored [26]. Note that CNNs do not need to recover original images to classify them correctly. They only need to reduce the impact of the perturbation to obtain a correct classification. Here, we explore filtering approaches that can reduce perturbation noise sufficiently to make them recognizable for CNNs. To evaluate robustness of k-RTIO, we applied different filters such as Blurring [100], Median Filter [50], Grayscale Filter and Gaussian Filter to reduce noise or distortion. As shown in Figure 3.18, none of these methods could improve image quality for DeepFace face recognition classifier significantly.

3.7 Discussion and Future Work

Adversarial perturbation is being studied as a way to address privacy concerns with scalable automated face recognition by CNNs [51; 105; 33]. In this work, we proposed some practical requirements for such defenses and proposed two perturbation schemes, UEP and k-RTIO, that meet those requirements. However, these schemes have their limitations.

Downside of Universal Perturbation: UEP generates a single transferable perturbation that can be used on multiple image to successfully fool an unknown CNN. The rationale for designing UEP, a universal transferable perturbation scheme, was to amortize the cost of learning a perturbation across multiple images, and

to be able to reverse the perturbation with a low storage cost of just storing one perturbation. However, the use of a single perturbation across multiple images opens up the possibility of a target CNN estimating or learning this perturbation by observing multiple perturbed images. Indeed, as described in Section 3.6.3, we show that it is possible to filter the perturbation sufficiently enough to allow correct classification. This leads us to observe that cost-efficient approaches to learn and store image-specific perturbations need to be developed before *learning-based* perturbation approaches can become practical defense tools against automated classification. Alternatively, if multiple independent universal perturbations can be computed and used, it may be possible to leverage them to thwart the perturbation estimation approach discussed in Section 3.6.3. Both these remain open problems.

Efficiency of Semantic Perturbations: In contrast to adversarial-learning approaches such as UEP, our semantic approach *k-RTIO* reduces the efficiency of a CNN in recognizing features, by reducing the similarity of adjacent pixels. *k-RTIO* is as a better choice than UEP, given that *k-RTIO* is both computationally cheaper and found to be more effective. We are encouraged by the feasibility to find cheaper techniques to thwart recognition by existing classifiers. While our initial evaluation shows *k-RTIO* is robust against common filtering techniques, it is not an exhaustive list and further work is needed in this direction.

Re-training a CNN on adversarial examples has been shown to improve the CNN’s robustness against such examples [74] and is a potential defense against *k-RTIO*. Although this approach is not computationally feasible for large scale

datasets [7], we did evaluate its effectiveness against k-RTIO using TinyImagenet dataset to train a CNN on both clean and k-RTIO perturbed images. Our evaluation showed that for small values of α , this robust CNN’s accuracy on perturbed images has not improved significantly ($< 30\%$ and 40% for $\alpha = 0.4$ and 0.45 , respectively).

Recognizability: Our findings indicate a sweet spot for k-RTIO parameters ($k = 3$, $\alpha = 0.45$, and $b < 20$) where classifiers have low success while the images remain visually recognizable (see Figure 3.12). However, it would be good to empirically evaluate the recognizability of *k-RTIO* perturbed images through user studies and to characterize the boundaries for human recognition in terms of parameter values for k , α , and b .

Lack of Strong Guarantees: Besides learning a robust CNN, online service providers may also be able to rely on other channels of information other than the image itself to infer information that can help them classify correctly, such as where, when, and with whom the image was shared. Further, an adversary may be able to use the structure present in overlay images to reduce their effort required to recreate the unique overlay image. While this can be done using visual clues manually, it may also be feasible to automate this although we are not aware of a specific way to efficiently achieve that. Similarly, it may be possible to find a smoothing filter that allows an adversary to correctly classify a k-RTIO image without reversing the perturbation. In other words, unlike traditional privacy mechanisms like encryption, perturbation-based defenses are not yet able to provide provable guarantees. This is especially the case as it is hard to quantify the information

leaked by perturbed images (especially since we require it to be sufficient enough for human recognition). At least in the case of ideal TPE schemes [123] the leaked bits can be succinctly characterized. While differential-privacy based perturbation approaches are emerging (*e.g.*, [29; 30]) they are currently not reversible. Perturbation schemes that thwart one pre-specified classifier while allowing another pre-specified classifier to classify correctly (*e.g.*, [51; 130]) have also been proposed. However, those approaches are also not reversible. Further, it is not clear how they fare against an unknown classifier. This lack of defined guarantees is not specific to just UEP or k-RTIO but seems to be true for adversarial perturbation based defenses in general. This is an open question that needs to be explored for image perturbation based privacy defenses to become practical.

Finally, while problems that are hard for AI and easier for humans have been explored in the context of distinguishing between humans and robots on the web (*e.g.*, CAPTCHAs), there seems to be a need to revisit such a paradigm for privacy against automated classifiers.

3.8 Conclusion

Given that users tend to store and share photos with friends and family using cloud-based platforms with weak access control options, the emergence of scalable and automated image classification is a serious threat to user privacy. In this chapter, we explored two reversible image perturbation techniques that can thwart classification and tracking by automated classifiers while at the same time allow-

ing recognition by humans. We showed that our k-Randomized Transparent Image Overlays (k-RTIO) approach is not only very effective but also computationally cheaper in contrast to learning-based perturbation methods. However, more work is needed before perturbation based approaches can truly become practical.

Chapter 4: Adversarial Examples against Super Resolution Convolutional Neural Networks

4.1 Introduction

Single-image Super-Resolution Convolutional Neural Networks (SRCNNs) [115; 73; 136] were originally designed to generate/recover a High Resolution (HR) image from its Low Resolution (LR) counterpart. Compared to traditional simple interpolation methods for generating high resolution images such as bilinear and bicubic interpolations [75], SRCNNs return high resolution images with better perceptual quality and more visually plausible details. Consequently, SRCNNs have become widely popular in a diverse range of applications including medical image analysis to improve diagnostic accuracy [107], satellite imaging for remote sensing and resource detection where it is important to distinguish between two or more similar objects [125], and surveillance applications [132]. While SRCNNs have legitimate and beneficial applications in different domains, they introduce privacy issues that need to be addressed. Previously, low resolution images were considered to provide some level of image privacy against recognition by either humans and classifiers [123; 46; 77; 129]. However, SRCNNs' ability to generate high resolution images from the low resolution ones for face recognition [103; 139; 21] poses a threat to previous approaches such as thumbnail-preserving encryption

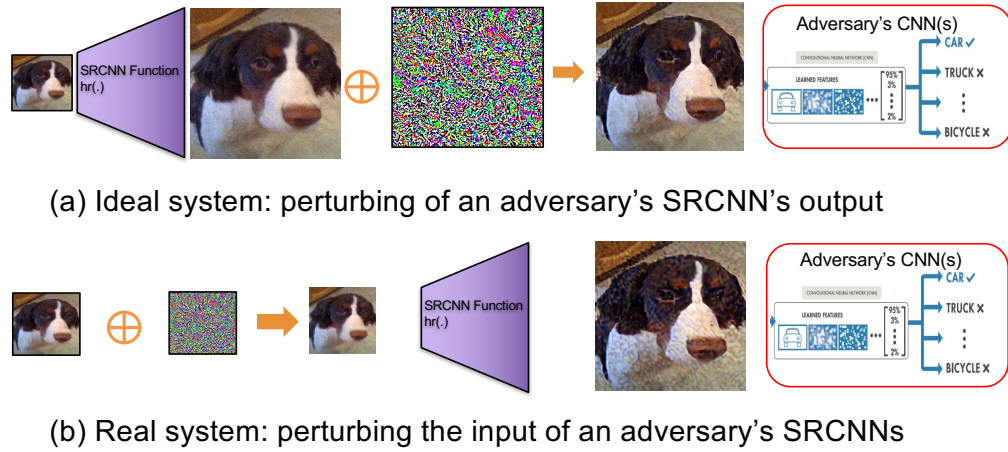


Figure 4.1: An adversary uses an SRCNN to generate HR images and gives it to an image CNN classifier for classification. (a) shows an ideal system in which the adversary's SRCNN is known and a user can add adversarial noise to generated HR by the SRCNN, (b) shows a real system in which the adversary's SRCNN is unknown and a user perturbs her LR images directly.

(TPE) [123; 129; 77] that leak low resolution thumbnails for usability purposes.

The privacy threat posed by AI-based face-recognition is real as evidenced by the emergence of image search applications such as Clearview AI Facial Recognition app [44], which can take a given picture of a person and search a database of more than 3 billion images scraped from Facebook, YouTube and millions of other websites to provide matching photos along with links to websites containing those photos. The emergence of powerful SRCNNs that can be utilized for face recognition from low resolution images exacerbates this threat.

Despite of high accuracy of CNNs, recently it has been shown that they are vulnerable to adversarial perturbations, and consequently adversarial perturbation has been explored as a promising image privacy protection mechanism against

automated face recognition by convolutional neural networks (CNNs) [51; 20; 28]. However, these approaches do not explicitly consider low-resolution images or take SRCNNs into account. Here, we aim to assess the survivability of the adversarial images trained on CNNs through SRCNNs

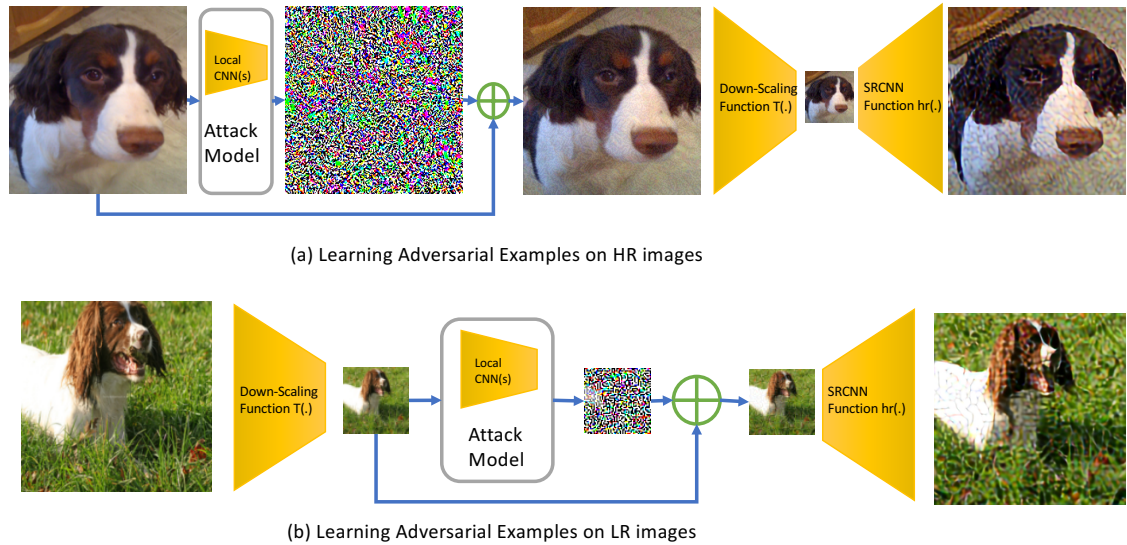


Figure 4.2: Two different approaches for generating low resolution adversarial examples for unknown SRCNNs using only a local CNN(s). In (a), an adversarial example is generated on a clean HR image and the down-scaled adversarial image is given to SRCNNs as input. In (b), an adversarial example is generated on an LR clean image and is given to SRCNNs as input. Target CNNs may or may not be known for users, but the target SRCNN(s) is unknown.

Contributions: In this chapter, we hypothesize and empirically show that adversarial perturbations targeting only CNN Classifiers can survive the processing by SRCNNs. We evaluate the survivability of both white-box and black-box adversarial perturbations through the state-of-the-art SRCNNs and demonstrate that joint optimization over both SRCNNs and CNNs is not necessary to learn low resolution

adversarial images (see Figure 4.1) in order to thwart target CNN classifiers. In summary, we make the following contributions:

- We hypothesize survivability of adversarial perturbations through SRCNNs.
- We assess the survivability of white-box and black-box adversarial perturbations. We then demonstrate that both (i) perturbations learned on LR, and (ii) down-scaled HR adversarial images can survive state-of-the-art SRCNNs.
- We demonstrate that a user can learn imperceptible noise for low resolution images on her small CNN(s) to make SRCNNs generate low quality high resolution images that lead CNNs to misclassify them.

The rest of this chapter is organized as follows: We describe our assumptions and the system model in Section 4.2 and discuss related work in Section 4.3. We introduce our methodology for generating LR adversarial images and evaluating their survivability through SRCNNs in Section 4.4. We finally present our results in Section 4.5. We discuss future work in Section 4.6 , and conclude in Section 4.7.

4.2 System and Adversary Model

In this chapter, we consider a situation where an adversary aims to classify low resolution images using their convolutional neural network classifiers. In order to improve their classification performance, the adversary passes the low resolution images through a super resolution convolutional neural network first, to generate a higher resolution image, and then uses a CNN classifier. Figure 4.1 depicts

such an adversary’s classification pipeline. We assume that the adversary’s image classification CNN is pre-trained (*e.g.*, using images scraped from the Internet; or using images stored or shared by users on online storage or sharing platforms).

Our goal is to help owners of the (low resolution) images (*e.g.*, users’ of social networks or image sharing platforms) thwart these unauthorized automated classification pipelines using image perturbation. Here the perturbation needs to be applied to the low resolution to cause SRCNNs to generate high resolution image that is misclassified by the CNN classifier.

However, it is unlikely that the image owners have any a prior knowledge of the SRCNNs and CNNs that may be used against their images in an adversary’s pipeline. Thus we assume that users do not have any knowledge about the SRCNN(s) that an adversary might use. For the image classification CNN, we consider the following two scenarios:

- A user does not have any knowledge about the adversary’s image classification CNN. This is the more likely case in real-world use. For example, in image sharing platforms, end users do not know what classifiers may be used by service providers or other curious users. This scenario requires perturbations for the images to be learned in a black-box setting.
- We also consider a scenario where we have access to the target CNN or can estimate the CNNs’ function by sending queries. In this case, perturbations for images can be learned using white-box adversarial learning techniques. While this scenario is not quite realistic, we do consider this to study whether

white-box perturbations have an advantage over black-box perturbations or vice-versa.

4.3 Related Work

We covered the significant works on image privacy in Chapter 3.2. Here we focus on approaches that learn adversarial noise for SRCNNs.

Recently, an attack model has been proposed for generating adversarial noise on low resolution images to decrease the quality of HR images generated by SRCNNs [23]. Similar to FGS attack for CNN classifiers, this method tends to learn adversarial noise for SRCNNs, but the objective function is to maximize the difference between a generated high resolution image and its original high resolution counterpart with minimum perturbation on the low resolution image as follows:

$$\min_{\delta} \|\delta\|_p - \mathbf{dist}(hr(I_{lr} + \delta) - I) \quad (4.1)$$

where δ , I , I_{lr} , hr , and \mathbf{dist} are adversarial noise, the original high resolution image, the low resolution image, an SRCNN's function which generates the high resolution of a given image and the distance function which measures the difference between two images, respectively. Peak signal to noise ratio (PSNR) or minimum square error can be used as distance function. This method assumes that the target SRCNN is known and accessible for the adversary. Moreover, the generated HR images from perturbed low resolution images are not necessarily able to fool

a CNN, since their goal is degradation of generated image’s quality, not fooling a CNN classifier. A joint optimization has been proposed for learning adversarial noise on low resolution images in [133], such that a target CNN will misclassify the generated high resolution images by a given SRCNN. This method finds an optimum adversarial noise for fooling a given CNN classifier. But, it assumes that both target SRCNN and CNN classifier are known.

4.4 Methodology

4.4.1 Overview

In this section, we first introduce our approaches for generating LR adversarial images and define the notation of survivability of adversarial images through SRCNNs¹. Then we introduce our approach for evaluating the survivability. Briefly, we will see the following topics in this section:

- In Section 4.4.2, we first introduce our approaches for generating low resolution adversarial images using a local CNN classifier(s). Images’ owners want to generate LR adversarial images which lead SRCNNs to generate HR adversarial images that are misclassified by the adversary’s classifier(s). As discussed in Section 4.2, the adversary only have access to LR images.

- In Section 4.4.3, we define survivability of adversarial images through SRC-

¹Here, we call generated adversarial examples with either white-box or black-box attack models adversarial images.

NNs and then introduce our approach for assessing that.

- Finally, in Section 4.4.4, we introduce our metrics used to assess survivability of adversarial images.

4.4.2 Generating Low Resolution Adversarial Images

As we discussed in Section 4.2, we assume that images' owners share Low Resolution (LR) images and an adversary uses an SRCNN(s) to generate High Resolution (HR) images to feed them to her CNN classifier(s). Therefore, the images' owners aim to perturb their LR images such that the adversary's SRCNN(s) generates adversarial HR images which will be misclassified by the adversary's CNN classifier. To generate adversarial LR images, we consider two different scenarios:

- Generating adversarial noise/perturbation on high resolution images and down-scaling them: as shown in Figure 4.2 (a), to generate LR adversarial images, one can learn adversarial noise/perturbation on a local CNN(s) for high resolution images, then down-scale the adversarial images. The down-scaled adversarial images will be passed through SRCNNs by the adversary to generate HR images for CNNs classifiers.
- Learning adversarial noise/perturbation on low resolution images: a user may only have a small set of images and may not be able to learn a large CNN. As shown in Figure 4.2 (b), for this case, one can train a small CNN(s) on LR images and learn adversarial noises/perturbations on that small local

CNN(s).

The adversarial images can be learned in a black-box or a white-box setting [36; 82; 84]. If a user has access to the adversary’s target CNN or can estimate the target CNNs [93] by sending queries, white-box adversarial examples are better choice since the generated noise is imperceptible for human eyes. However, adversarial perturbations are black-box attacks and they are more suitable (compared to white-box adversarial noise) for image privacy, since they are designed to fool unknown CNNs with high probability .

4.4.3 Survivability Evaluation

There are several adversarial learning methods for generating adversarial images [37; 36; 18; 83]. Here, we aim to evaluate the survivability of adversarial images through SRCNNs. A low resolution adversarial image survives through SRCNNs if its generated HR image has the same properties as the original HR adversarial image. Adversarial images have two nice properties: (i) the ability to fool CNNs, and (ii) preserving the true class of adversarial image ². To evaluate the ability of fooling CNNs for generated high resolution images by SRCNNs, we use misclassification rate as transferability metric. For assessing the perceptual survivability of generated high resolution images from low resolution adversarial images, we use the similarity metrics which quantify the quality of a reconstructed image from its low resolution image using SRCNNs. The similarity metrics compare the quality of

²The adversarial images belong to the class of original clean HR image perceptually

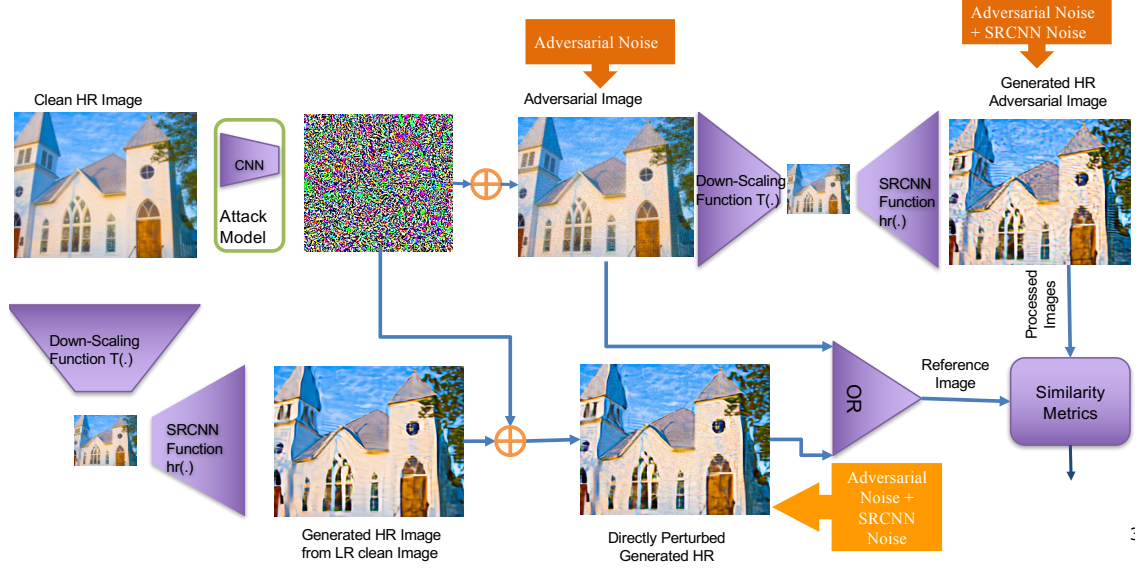


Figure 4.3: Two approaches of selecting a reference image for similarity metrics: (i) The original HR adversarial image, and (ii) Directly perturbed generated HR image.

the generated high resolution with a reference image. We consider two different scenarios for selecting the reference image to evaluate the perceptual quality of a generated high resolution images by SRCNNs as follows:

- Original HR adversarial perturbed image as reference image: SRCNNs tend to reconstruct the original high resolution image of a given input. The original HR adversarial image is the target image that a user wants an SRCNN to generate from its down-scaled. Therefore, the original HR image (see equation 4.2) can be used as a reference image.

$$\text{Sim}(hr_k(T(I + \delta)), I + \delta) \quad (4.2)$$

where I , δ , hr_k , and T are a benign image, adversarial perturbation, k^{th}

SRCNN function which returns the high resolution of a given image and down-scaling function, respectively. To measure the similarity of two images, we utilize different similarity metrics ($\text{Sim}(\cdot, \cdot)$) that are introduced in Section 4.4.4.

- Directly perturbed high resolution image generated from a clean image as reference image: Even though SRCNNs aim to generate the exact high resolution of the given input, the generated high resolution image is different from its original high resolution counterpart, and usually has a small noise. The similarity between directly perturbed high resolution image generated from a clean image ($hr_k(T(I)) + \delta$) and generated high resolution image from LR adversarial image ($hr_k(T(I + \delta))$) (see equation 4.3) gives better estimation of the survivability of adversarial noise/perturbation, since both carry a SRCNN’s noise. In other words, the noise added by an SRCNN is not considered in the similarity estimation.

$$\text{Sim}(hr_k(T(I)) + \delta, hr_k(T(I + \delta))) \quad (4.3)$$

where I , δ , hr_k , and T are the clean image, adversarial perturbation, k^{th} SRCNN function which returns the high resolution of a given image and a down-scaling function, respectively.

4.4.4 Metrics

In this section, we define the transferability metric and introduce the similarity metrics of Peak Signal to Noise Ratio (PSNR), Structural Similarity Index [65] and perceptual similarity [135].

Transferability (TR): To measure the transferability of generated high resolution images, we consider the misclassification rate which is defined as follows:

$$TR = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\operatorname{argmax} F(hr_k(I'_i)) \neq Y_{I'_i}^*) \quad (4.4)$$

where I'_i , $hr_k(I'_i)$, and $Y_{I'_i}^*$ are the i^{th} perturbed image, the high resolution image of generated by k^{th} super resolution convolutional neural network (SRCNN) and the true label of the perturbed image, respectively. Also, $F(x)$ returns the probability vector generated by a CNN classifier for image x and $\mathbb{I}(t)$ is the identity function which returns one when t is true, and zero otherwise.

Peak Signal to Noise Ratio (PSNR): This metric uses the normalized Minimum Square Error (MSE) between two images. Unlike MSE that depends on the scale of pixels' value of images, PSNR considers the ratio of the maximum possible value for a pixel to difference between pixels as described below:

$$\begin{aligned}
 MSE &= \sum_{i,j}^{M,N} \frac{[I_1(i,j) - I_2(i,j)]^2}{M * N} \\
 PSNR &= 10 \log_{10} \frac{R^2}{MSE}
 \end{aligned}
 \tag{4.5}$$

where I_1 and I_2 are two reference and reconstructed images, respectively, M and N are the resolution (number of rows and columns) of the images and R is the maximum pixel's values. For two identical images MSE converges to 0 and therefore PSNR value converge to a large value. In other words, if two images' pixels value are too close, their PSNR would have a very large value.

Structural Similarity Index(SSIM): Unlike PSNR, SSIM is based on visible structures in the image, since human visual perception depends on structural information existing in an image and not pixel's values alone. Therefore, this metric focuses on local pattern of pixel intensities that have been normalized for luminance and contrast. SSIM metric returns a value between $[0, 1]$ and for two identical images it returns value of one, and by increasing the difference between two images, it returns smaller values.

Perceptual Similarity (PerSim): Both PSNR and SSIM are susceptible to noise and a small amount of noise could cause a significant degradation on these metrics, while the images would not change perceptually. To address this problem, perceptual similarity was proposed in [135], which measures the similarity not in

the image space (similarity between pixel values), but rather in a feature space. Feature space is the output of intermediate layers of a CNN. This metric considers two images similar if for a given CNN they generate similar values in feature space. This metric returns a value in the range of $[0, 1]$ and for two identical images it returns zero, by increasing the difference between images, it returns a larger value.

In this section, we introduced our methodology for generating LR adversarial images and how we evaluate the survivability of them through SRCNNs. We present our evaluation results in Section 4.5.

4.5 Evaluation

In this section, we evaluate the survivability of adversarial images through SRCNNs for two different attack models of white-box and black-box attacks and for both methods of generating LR adversarial images (See Figure 4.2). We choose two state-of-the-art black-box attacks to generate adversarial perturbations; Universal Ensemble Perturbation (UEP) proposed in Section 3.5 designed for image privacy and Universal Perturbation (UP) [83] designed for fooling unknown CNNs. We also use Fast Gradient Sign (FGS) to generate adversarial images with imperceptible noise. Compared to adversarial perturbation generation methods (*e.g.*, UEP and UP), FGS has less transferability but it creates imperceptible noise for LR images. Table 4.1 shows different scenarios we consider for our evaluations. For instance, we use FGS for generating white-box HR adversarial images and black-box LR

adversarial images on a local small CNN³.

	Learning LR Adversarial Images	Down-Scaling HR Adversarial Images
Black-Box	FGS (Section 4.5.3) & Universal Perturbation (Section 4.5.2)	Universal Ensemble Perturbation (Section 4.5.2) & FGS (Section 4.5.3)
White-Box	NA	FGS (Section 4.5.3)

Table 4.1: Evaluation scenarios. We consider two attack models of white-box and black-box attacks with different approaches of generating LR adversarial images.

4.5.1 Simulation Setup

SRCNNs: We select four state-of-the-art super resolution convolutional neural networks of (i) RCAN [69; 136], (ii) CAR [115] which obtained the best PSNR, (iii) SPSR [73] whose objective function minimizes perceptual similarity, and (iv) deep face super-resolutions [72] which is trained only on faces. SPSR is trained for scale of 4⁴ and Deep Face SRCNN is trained for scales of 4 and 8. Both CAR and RCAN support scales of 2, 4 and 8.

Datasets: We consider two different datasets, (i) a set of 1000 celebrities images

³Note that here we assume an adversary has a large CNN trained on HR images. Therefore, we do not have white-box attacks for Learning LR adversarial images.

⁴An SRCNN with scale of 4 generates a HR image with resolution of $4M \times 4N$ from a LR image with resolution of $M \times N$.

sampled from Facescrub dataset [88], and (ii) a set of 1000 images sampled from ImageNet dataset [102]. ImageNet dataset contains more than 1000 classes and 1M images. For our evaluations, we select 1000 images from the 10 easiest classes for classifications which are less sensitive to noises and consequently, more difficult to misclassify.

Classifiers: To measure the transferability of the adversarial noise and perturbation, we consider a state-of-the art online face recognition model for celebrities called `clarifai.com` and pre-trained ImageNet classifiers trained on ResNet101 CNN and ResNet152 CNN [42]. Also, to train the adversarial noise on low resolution images, we train a small CNN with VGG-11 [109] structure on the 10 easiest classes of ImageNet dataset. Samples from these classes are less sensitive to noise and are more difficult to misclassify.

4.5.2 Adversarial Perturbations

Here, we evaluate the survivability of two different adversarial perturbation methods of universal perturbation and universal ensemble perturbations. Both of these methods generate transferable perturbations at the cost of adding more noise. We first evaluate the universal ensemble perturbation learned on different resolutions specially for image privacy. Then we assess the universal perturbation learned on LR images on a small CNN with 10 classes only.

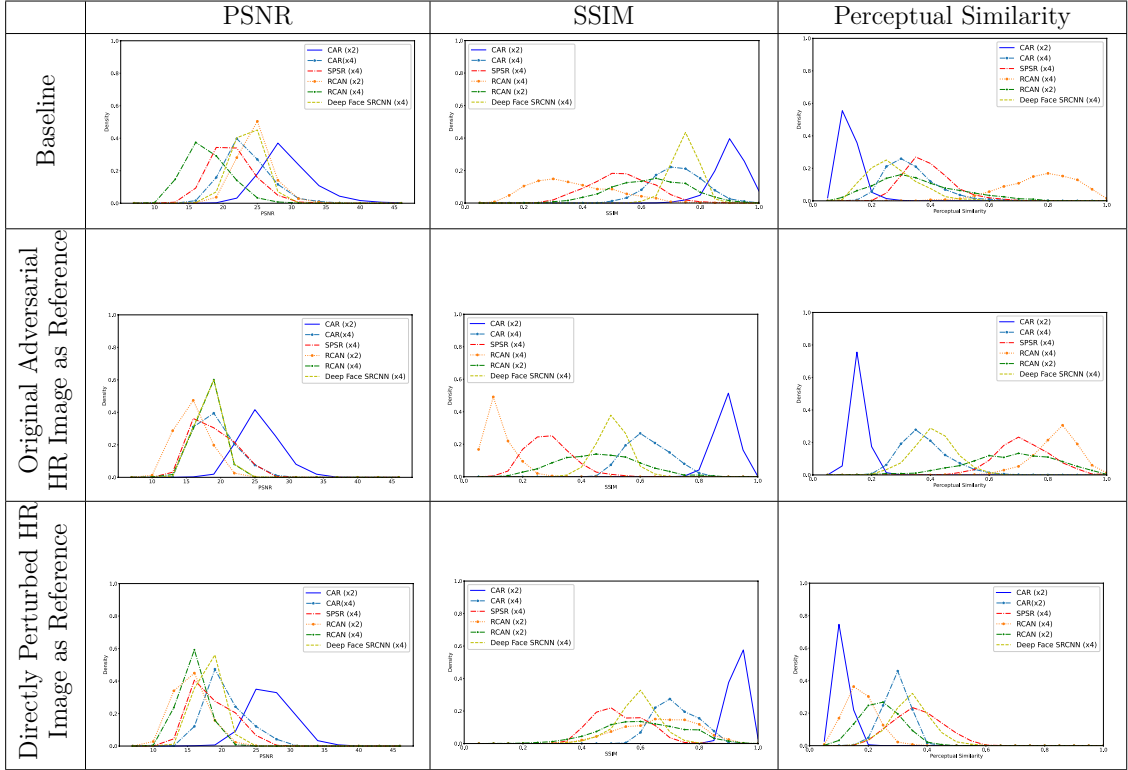


Figure 4.4: Each column shows the density distributions for a similarity metric to compare (i) the generated high resolution images from down-scaled clean images and original HR images ($\text{Sim}(I, hr_k(T(I)))$), (ii) the generated high resolution images from down-scaled UEP perturbed images and original UEP perturbed images ($\text{Sim}(I + \delta, hr_k(T(I + \delta)))$), and (iii) the generated high resolution images from down-scaled UEP perturbed images and the directly UEP perturbed high resolution images generated from down-scaled clean images ($\text{Sim}(I + \delta, hr_k(T(I) + \delta))$). Sim function is a similarity function, PSNR, SSIM or perceptual similarity (PerSim). Two more similar images return higher values for PSNR and SSIM and lower value for perceptual similarity.

Universal Ensemble Perturbation (UEP): To evaluate the survivability of UEP through SRCNNs, we use Facescrub dataset since this perturbation is designed for image privacy and is learned on faces. For misclassification rate, we use the `clarifai.com` celebrities face recognition model.

Baseline: Before evaluating the survivability of UEP through SRCNNs, we assess the performance of SRCNNs on generating recognizable high resolution images from clean LR images for the target classifier, `clarifai.com` face recognition model. So we use the 1000 sampled face images from Facescrub dataset that are all recognizable for `clarifai.com` and give the down-scaled images to SRCNNs. Table 4.2 presents the accuracy of `clarifai.com` on high resolution images scaled up by SRCNNs for different scales of 2 and 4. SRCNNs could not generate classifiable high resolution images for larger scale 8. As shown, 86.16% of LR images can be reconstructed (using CAR (x4)) such that they are recognizable for the celebrity face recognition model.

SRCNNs (scale)	Baseline		UEP Transferability	
	Face Detection	Face Recognition	Face Detection	Face Recognition
CAR (x2)	100%	87.44%	99.8%	1.72%
CAR (x4)	100%	86.16%	99.75%	0.5%
RCAN (x2)	99.26%	71.74%	99.8%	1.61%
RCAN (x4)	88.73%	33.05%	100%	0.4%
SPSR (x4)	100%	69.25%	99.65%	0.129%
Deep Face SRCNN (x4)	100%	71.82%	99.61%	1.66%

Table 4.2: Accuracy of `clarifai.com` celebrity face detection and recognition models for generated high resolution images from clean images (Baseline) and generated high resolution images from down-scaled UEP perturbed images.

We also use 3 similarity metrics of PSNR, SSIM and perceptual similarity introduced in Section 4.4.4 to assess the quality of generated high resolution images by SRCNNs. As shown in 4.4, we measure the density for each similarity metrics. The first row in Figure 4.4 shows the similarity between an original high resolution image and the generated high resolution counterpart. CAR (x2), Deep Face SRCNN and CAR (x4) generate more similar images to original ones. These

results are compatible with the face recognition model’s accuracy presented in Table 4.2. In other words, these three SRCNNs generate more recognizable images for `clarifai.com` face recognition model. As we expected, CAR (x2) obtained larger PSNR and SSIM values and smaller values for perceptual similarity. For the scale of 8, CAR, RCAN and Deep Face SRCNNs could not generate recognizable images for the face recognition model.

Survivability of UEP: To evaluate survivability of UEP through SRCNNs, we perturb original high resolution images using UEP with $\beta = 3$ (as suggested in Section 3.6.1). To generate perturbed LR images, we down-scale UEP perturbed images by getting average of each block and use the state-of-the-art SRCNNs to generate high resolution images.

As shown in Table 4.2, at least 98%(1 – 1.79) of the generated HR images from down-scaled UEP perturbed images can fool `clarifai.com` face recognition model successfully. To measure survivability, we consider three different similarity metrics of PSNR, SSIM and perceptual similarity. As shown in Figure 4.4, CAR (x2) and RCAN (x2) generate more similar HR images to benign perturbed HR images. For example, the perceptual similarity between generated high resolution images by CAR (x2) from down-scaled UEP perturbed images, and the original perturbed images is less than 0.15 for more than 75% of them (see the figure in second row and the last column).

As discussed, the generated high resolution images by SRCNNs are not exactly the same as their benign HR images and they contain small amount of noise which

leads the similarity metrics to degrade. Therefore, to assess only the survivability of an adversarial noise/perturbation, we consider the similarity between directly perturbed HR images generated from clean LR images (as reference image) and generated HR images from low resolution of adversarial images. As shown in the last row of Figure 4.4, when we use directly perturbed generated HR image from low resolution clean as reference image, the similarity metrics' values improve. For example, most of perceptual similarity values for CAR (x4) are less than 0.3 when the directly perturbed HR images are used as reference images, while the original adversarial images are used as reference images (in the second row), most of perceptual similarity values for CAR (x4) are around 0.4.

Universal Perturbation: To evaluate the survivability of universal adversarial perturbation in black-box setting, we train a small CNN on the 10 easiest classes on low resolution images (56×56). We train a universal perturbation with $\epsilon = 0.03$ which was able to fool the small CNN for 80.1% of images. Then we use CAR (x4) to generate high resolution images for ImageNet classifiers (Both ResNet101 and ResNet152). The high resolution images could fool ImageNet classifiers of ResNet101 and ResNet152 for 92.4% and 91.8% of the times, respectively. Here, since we learn an adversarial perturbation for LR images, we use generated HR from clean LR images as reference images. In this case, we do not have original HR adversarial images for comparison. The average (std) of PSNR, SSIM and perceptual similarity are 12.3(1.66), 0.325(0.139), 0.408(0.104), respectively. Since we consider generated HR from LR clean images as reference images, PSNR

and SSIM similarity metrics have lower values and perceptual similarity has higher value. Note that the smaller values are better for perceptual similarity.













	Clean	FGS Attacks		
ϵ	0	0.01	0.03	0.05
LR Image				
Original HR Image				
Generated HR				

Figure 4.5: The first row shows the low resolution of the second row images and the third row shows generated HR images of low resolution images shown in the first row by CAR(x4). The first column of images shows the clean images and the rest column shows the adversarial examples generated for benign high resolution images with FGS attack and different ϵ values. I , δ , $T(\cdot)$ and $hr(\cdot)$ are the original image, adversarial noise, down-scaling function and SRCNN function which returns high resolution of a given images, respectively.

4.5.3 Adversarial Examples

There are many models for generating adversarial examples which can fool a known target CNN with imperceptible noise. To evaluate the survivability of adversarial

examples through SRCNNs, we use the samples taken from the 10 easiest class of ImageNet dataset. As discussed in Section 4.4.3, we consider two different scenarios for generating low resolution adversarial examples: (i) generating adversarial examples on high resolution images and down-scaling them, (ii) learning adversarial noise on low resolution images using a small CNN(s) trained on low resolution images.

Before evaluating adversarial examples, we assess the quality of generated high resolution images by SRCNNs for the ImageNet CNN classifier. For this, we use low resolution version of these images as input to SRCNNs and give the generated HR images to ImageNet classifier. Our experiments show that the generated high resolution images by CAR (x2), RCAN (x2) and CAR (x4) have 86%, 69.9% and 39% accuracy on the ImageNet ResNet101 CNN classifier, respectively, and SPSR (x4) and RCAN (x4) could not generate classifiable images. We choose the images that are classified correctly and learn adversarial examples on them, then we evaluate them on SRCNNs. We describe our results for each scenario of generating low resolution adversarial examples below:

Survivability of Down-Scaled Adversarial Examples: To generate adversarial examples, we select the images classified correctly by each SRCNN to learn adversarial noise using Fast Gradient Sign (FGS) attack on ImageNet classifier trained on ResNet101 for different values of step size ($\epsilon \in \{0.001, 0.01, 0.02, 0.04, 0.05\}$). We measure transferability for both white-box attack in which the target CNN is ImageNet classifier trained on ResNet101 and black-box attack in which the target

CNN is ImageNet classifier trained on ResNet152. We also consider transferability rate for two different HR adversarial images namely (i) generated HR adversarial images from down-scaled HR adversarial examples (when SRCNNs are not known), and (ii) directly perturbed generated HR images from LR cleans (when SRCNNs are known). As shown in table 4.3, for black-box attacks the transferability is lower, since adversarial example generation methods create less perturbation at cost of losing transferability. Also, the transferability of generated HR adversarial images from down-scaled HR adversarial examples (when SRCNNs are not known) is very close to transferability of directly perturbed generated HR images from LR cleans. In other words, it shows having access to SRCNNs to directly perturb their output does not increase the transferability.

To measure the survivability, we use similarity metrics and consider two different scenarios for calculating similarity metrics (i) the original adversarial examples as reference images (ii) directly perturbed HR images generated from clean low resolution images as reference images.

Original adversarial example as reference image: As shown in Table 4.4, for the small values of ϵ , the adversarial noise does not transfer well. However, the larger ϵ values have higher transferability (misclassification rate) but lead to more distortion in generated high resolution images even though the adversarial noise in low resolution images is imperceptible. (see Figure 4.5). Consequently, larger values of ϵ lead the similarity metrics to degrade. For example, the adversarial images generated for $\epsilon = 0.01$ and $\epsilon = 0.03$ (the second row in Figure 4.5) are perceptually

		Black-Box (ResNet152)		White-Box (ResNet101)	
	ϵ	TR_R	TR_I	TR_R	TR_I
CAR (x4)	0.001	20.94%	20.65 %	15%	15.63%
	0.01	29.2%	24.78 %	28.31%	19.76%
	0.02	43.95%	26.55%	42.3%	26.54%
	0.03	56.63%	28.02%	58.7%	28.61%
	0.04	66.37 %	29.45%	69.32%	31.56%
	0.05	75.51 %	31.27%	76.7%	37.46%
CAR (x2)	0.01	27.86%	16.9%	38.32%	23.96%
	0.02	39.05 %	23.96%	52.43 %	38.81 %
	0.03	50.85%	28.47%	59.97%	47.32%
	0.04	57.29%	35.89 %	64.84%	52.92%
	0.05	64.72%	41.24%	69.09%	57.42%
RCAN (x2)	0.01	50.33%	48.82%	55.03%	55.87%
	0.02	66.78 %	66.61%	68.79%	71.14%
	0.03	57.05 %	56.71%	57.88%	62.08%
	0.04	68.28 %	66.94%	70.80%	69.46%
	0.05	77.68 %	77.18%	77.18%	77.34%

Table 4.3: Transferability of generated HR from down-scaled adversarial examples (TR_R) and transferability of directly perturbed HR images generated from LR clean images (TR_I) for both black-box (ResNET152) and white-box setting (ResNET101)

similar but the generated high resolution images (the third row in Figure 4.5) from their low resolution images are different (the perceptual similarity value of CAR (x4) for $\epsilon = 0.01$ and $\epsilon = 0.03$ are 0.276 and 0.446 in average).

Directly perturbed HR image generated from an LR clean sample as a reference image: SRCNNs do not generate the exact high resolution image of a given input and add small amount of noise. Therefore, we also consider the similarity between the directly perturbed high resolution images generated from LR clean images ($hr(T(I)) + \delta$) as references images) and the generated high resolution im-

		Original HR adversarial Image as reference			Directly Perturbed HR Image as reference		
	FGS(ϵ)	PSNR (avg (std))	SSIM (avg (std))	PerSem	PSNR(avg (std))	SSIM(avg (std))	PerSim (avg (std))
CAR (x2)	0.01	26.65 (2.97)	0.869 (0.034)	0.1 (0.024)	33.07 (2.09)	0.946 (SSIM)	0.027 (0.018)
	0.02	25.84 (2.34)	0.82 (0.03)	0.12 (0.024)	28.92 (1.46)	0.87 (0.04)	0.055 (0.024)
	0.03	24.77 (1.83)	0.78 (0.034)	0.143 (0.029)	26.48 (1.16)	0.82 (0.047)	0.09 (0.035)
	0.04	3.64 (1.45)	0.742 (0.038)	0.167 (0.034)	24.65 (0.98)	0.775 (0.05)	0.124 (0.042)
	0.05	22.53 (1.166)	0.712 (0.04)	0.189 (0.039)	23.16 (0.84)	0.7396 (0.05)	0.156 (0.047)
CAR (x4)	0.001	20.22 (3.23)	0.677 (0.105)	0.231 (0.059)	47.9 (2.76)	0.999 (0.0004)	0.0008 (0.0005)
	0.01	20.07 (3.12)	0.638 (0.086)	0.276 (0.056)	29.41 (2.23)	0.91 (0.023)	0.067 (0.041)
	0.02	19.62 (2.77)	0.564 (0.056)	0.366 (0.079)	325.22 (1.74)	0.785 (0.058)	0.1762 (0.089)
	0.03	19.02 (2.36)	0.50 (0.047)	0.446 (0.096)	22.88 (1.39)	0.688 (0.079)	0.273 (0.114)
	0.04	18.36 (1.98)	0.457 (0.047)	0.51 (0.104)	21.18 (1.146)	0.616 (0.088)	0.353 (0.12)
	0.05	17.69 (1.66)	0.422 (0.048)	0.562 (0.11)	19.83 (0.97)	0.56 (0.09)	0.418 (0.132)
RCAN (x2)	0.01	18.73 (4.41)	0.61 (0.153)	0.356 (0.186)	24.44 (8.82)	0.85 (0.159)	0.072 (0.077)
	0.02	16.58 (4.22)	0.552 (0.164)	0.446 (0.2)	19.79 (6.47)	0.778 (0.15)	0.0958 (0.066)
	0.03	18.68 (4.69)	0.572 (0.164)	0.333 (0.2)	22.69 (6.3)	0.852 (0.094)	0.092 (0.07)
	0.04	17.63 (4.3)	0.55 (0.155)	0.35 (0.17)	19.1 (5.78)	0.7 (0.17)	0.147 (0.09)
	0.05	15.39 (3.41)	0.47 (0.134)	0.41 (0.173)	16.64 (4.68)	0.649 (0.16)	0.183 (0.086)

Table 4.4: Similarity metrics for evaluating adversarial examples survivability through SRCNNs. We measure PSNR, SSIM and perceptual similarity for generated HR images from down-scaled of adversarial examples for two different cases of (i) original adversarial examples as reference images ($\text{Sim}(hr(T(I + \delta)), I + \delta)$), and (ii) directly perturbed HR images generated from clean low resolution images ($\text{Sim}(hr(T(I)) + \delta, I + \delta)$) as reference images.

ages from the down-scaled adversarial examples ($hr(T(I + \delta))$). Since both have SRCNN’s noise, the similarity between these images are higher. For example for $\epsilon = 0.03$ and CAR (x4), the average of perceptual similarity improves to 0.273 from 0.446.

Survivability of Directly Generated Low Resolution Adversarial Examples: As discussed in Section 4.4, we want to evaluate if one can train a small CNN on low resolution images and learn adversarial noises which can survive through SRCNNs. To this end, we train a small CNN (VGG-11) on low resolution images of the 10 easiest classes of ImageNet. To learn a small CNN, we reduce the resolution of the

images to 56×56 and use CAR (x4) as a target SRCNN. Here, we assume that an adversary has a large ImageNet classifier (the one with 1000 classes with input size of 224×224) and she wants to classify the users' images.

We select 300 low resolution images for which ImageNet classifier classifies their generated high resolution images by CAR (x4) correctly. We then learn FGS attack for different $\epsilon (\in \{0.01, 0.015, 0.02, 0.03, 0.04, 0.05\})$ values on those low resolution images. To measure the survivability, we evaluate the misclassification rate on the ImageNet classifier (ResNet101) and the similarity between high resolution images generated from clean images as reference images and high resolution images generated from adversarial examples. As shown in Figure 4.6, the adversarial noise in LR resolution images are imperceptible to human eyes even for large ϵ values, while it causes the SRCNNs to generate adversarial perturbed HR images which are misclassified by a CNN classifier.

ϵ	TR	PSNR	SSIM	PerSim
0.01	88.8%	30.34(1.37)	0.86(0.04)	0.145 (0.079)
0.15	90%	27.24 (1.05)	0.744 (0.07)	0.24 (0.10)
0.02	92.8%	25.02 (0.85)	0.64 (0.1)	0.32 (0.11)
0.03	95.8%	21.83 (0.67)	0.51 (0.11)	0.42 (0.11)
0.04	99.4%	19.57 (0.63)	0.413 (0.11)	0.48 (0.1)
0.05	99.7%	17.85(0.65)	0.35(0.1)	0.52 (0.09)

Table 4.5: Evaluating the survivability of white-box attacks learned on low resolution images for CAR (x4) SRCNN. The first column shows the ϵ value for FGS attack. The second column shows transferability of high resolution images generated from low resolution adversarial examples and the rest of columns show similarity between the HR generated from clean LR image and the generated HR image from its LR resolution adversarial example.

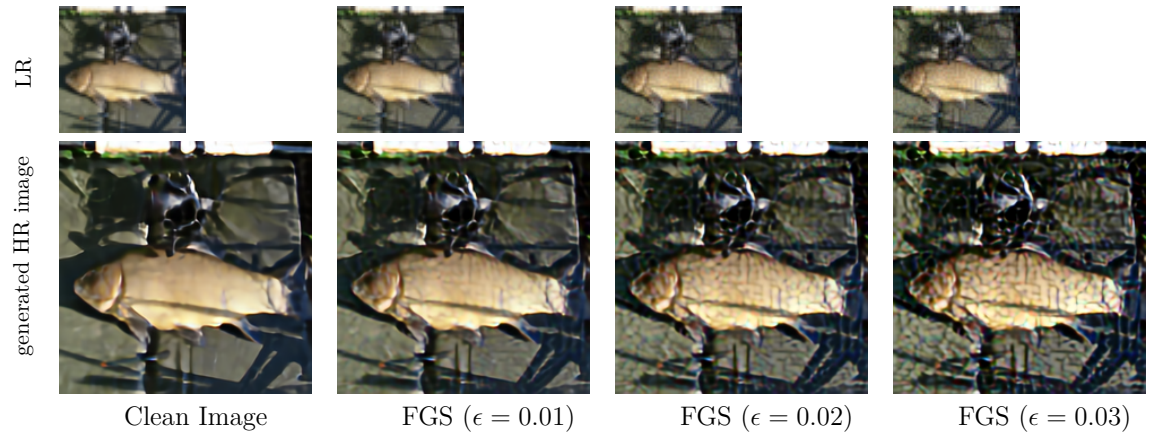


Figure 4.6: The first column shows a clean low resolution image and its high resolution images by CAR (x4) and the rest columns show the adversarial examples generated on low resolution images and their high resolution images by CAR (x4). As shown by increasing the ϵ values leads the adversarial noise to increase and the quality of generated high resolution images to decrease.

As shown in Table 4.5, for even small values of $\epsilon (= 0.01)$, the adversarial noise learned on small CNN is transferred well through CAR (x4) and fooled the the larger CNN classifier for more than 88% of the images. In Figure 4.6, the high resolution images generated from LR adversarial examples with $\epsilon = 0.01$ have imperceptible noise and large values for PNSR and SSIM metrics, and a low value for perceptual similarity metric. Note that the small values for perceptual similarity metric and large values for SSIM show that the high resolution images generated from adversarial examples are very similar to the high resolution images generated from clean samples.

4.6 Future Work

In this chapter we showed that one can learn adversarial images for SRCNNs using only a local CNN. Here we introduce some interesting directions for future work.

Survivability of Other Adversarial Attack Models: In this chapter, we only considered FGS attack to generate adversarial examples, since this method is computationally cheap. For future work, one could consider exploring other adversarial models, like C&W attack [18] and DeepFool attack [82]. These methods can generate smaller amount of noise and have higher transferability compared to FGS model.

Survivability of Random Noises: In this chapter, we focused on image privacy against automated image classifiers. However, there are other kind of privacy concerns like minimizing the chances of identifying a single record in a release of a large databases, called differential privacy. Random noise like Laplacian noise can provide measurable image privacy guarantee for example for differential privacy. Differential privacy is a popular privacy concept. Evaluating the survivability of random noise that guarantees differential privacy can be a valuable future work direction for this work.

Evaluating Against Robust CNNs: It has been empirically shown that adversarial images can fool unknown CNNs but there is no guarantee that they have the same transferability on robust CNNs. There are many approaches tend to make CNN robust by learning them on adversarial examples. These approaches aim

to train robust CNNs with ability of classifying adversarial examples correctly. Even though these approaches are still vulnerable to adversarial perturbations and computationally expensive [17], evaluating the generated HR resolution images from LR adversarial images can be a valuable direction as future work.

4.7 Conclusion

In this chapter, we empirically showed that adversarial noise/perturbations learned only on CNNs can lead super resolution convolutional neural networks to generate adversarial high resolution images which fool CNN classifiers. We evaluated both with-box adversarial noise and black-box adversarial images' survivability through SRCNNs. We showed that even for imperceptible white-box noise trained on a small local CNNs trained on limited images, the quality of generated high resolution images by SRCNNs degrades significantly.

Chapter 5: Detection and Rejection of Adversarial Examples and Out-Distribution Samples

5.1 Introduction

Convolutional Neural Networks (CNNs) have become popular due to their high accuracy for image and video analysis. Despite their strong performance, it has been demonstrated that they are highly susceptible to adversarial examples, especially when dealing with high dimensional inputs such as images [121; 119]. An adversarial image as previously discussed is one that has been perturbed with a noise signal designed to fool the CNN. While CNNs confidently misclassify such adversarial examples¹, they are perceptually similar to the original image and easily recognizable by humans (See Figure 5.1).

Many algorithms for generating adversarial examples have been proposed (*e.g.*, [38; 59; 18; 85]). Those algorithms can broadly be classified as either white-box or black-box attack algorithms. In white-box attacks, an adversary knows the target classifier’s exact model parameters and learns adversarial examples over it. In contrast, in black-box attacks, an adversary does not have any knowledge about the target classifier. Therefore, in a black-box attack setting, an adversary learns adversarial examples over a local CNN without access to the target classifier.

¹We use the terms adversarial image and adversarial example interchangeably.

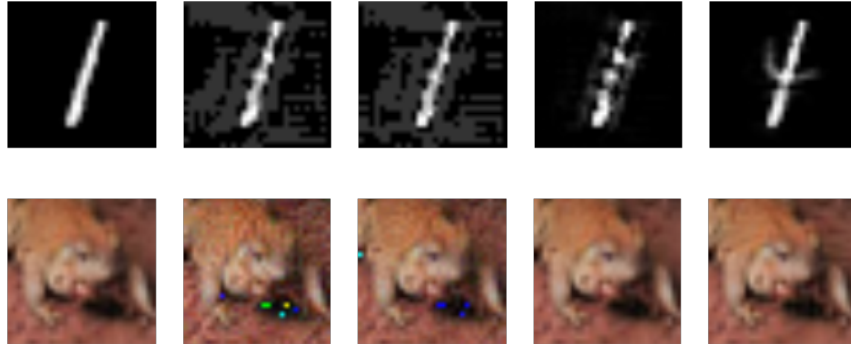


Figure 5.1: Adversarial examples for MNIST (first row) and CIFAR-10 (second row). From left to right: original images, T-FGS, FGS, DeepFool, and C&W (L_2) adversarial examples

However, due to the transferability [121] of black-box adversarial images, target CNNs can still be highly vulnerable to them [121; 43].

Furthermore, another serious challenge for CNN-based systems is that when a test sample comes from a different concept or class that is not part of the training set (*i.e.*, in-distribution samples), then CNNs assign them to one of the predefined classes they are trained on, possibly with high confidence. We call such samples as natural (*i.e.*, not synthetically generated) out-distribution samples as they are semantically and statistically different from the in-distribution samples. From a decision-making perspective, this is of great concern as CNNs show confidence that is clearly inappropriate, particularly for security sensitive tasks. This behavior demonstrates that neural networks do over-generalization in some regions that are empty of in-distribution samples (*i.e.*, belonging to out-distribution samples). In general, there are two approaches of (i) training robust CNNs, and (2) adding a post-processing step to existing CNNs, to detect and reject such examples. The

first approach tends to revise the decision boundaries of CNNs by adding auxiliary training data, while the second approach tends to make the current CNNs robust without re-training them. In this chapter, with the aim of addressing these challenges, we propose two different methods of (i) Augmented CNNs trained on a small set of auxiliary data (out-distribution samples and interpolated samples) which follows the first approach and, (ii) Adversarial profiles, a post-processing mechanism for adversarial and out-distribution examples detection.

In brief, our contributions are as follow:

- We propose to add an additional *dustbin* class containing (i) *natural out-distribution samples* (*i.e.*, natural samples that are statistically and semantically different from in-distribution samples), and (ii) *interpolated in-distribution data* (created by interpolating selected pairs of in-distribution samples from two different classes) to train an augmented CNN . This approach:
 - is a simple yet effective approach to reduce misclassification rate of CNNs for black-box adversarial examples by adding a *dustbin* class to learn a better feature space.
 - unlike previous approaches, i) does not need access to adversarial examples for training, ii) does not require additional networks (*e.g.*, CNNs, autoencoders *etc.*) to detect adversarial examples, and iii) does not noticeably sacrifice accuracy on clean samples.
 - is robust against well-known attacks (see Chapter 2 for details of at-

tacks and Section 5.4 for our results) and can either reject or classify adversarial examples correctly $> 95\%$ of the time for MNIST and $> 75\%$ of the time for CIFAR-10 on average.

- We propose to add a post-processing detection mechanism that uses a set of universal perturbations called adversarial profiles:
 - to make existing CNNs robust against out-distribution examples without retraining them or requiring any out-distribution or adversarial samples for learning.
 - which makes the MNIST CNN robust against out-distribution samples and can detect adversarial examples for this dataset at-least $> 59\%$ of the time.

The rest of this chapter organized as follows: In Section 5.2, we discuss related work and in Section 5.3, we introduce and evaluated our proposed method of Augmented CNNs to learn robust CNNs and in Section 5.5, we introduce and evaluate our Adversarial Profile method, a post-processing approach for detection and rejection of suspicious inputs. Finally we discuss future work and conclude in Section 5.6 and Section 5.7, respectively.

5.2 Related Work

It has been shown that adversarial examples are easy to generate and they are very successful in fooling even CNNs whose parameters are not known to the adversary

(*i.e.*, black-box attacks) [121; 58; 36]. Here, we introduce works that address adversarial examples in image classification for both white-box and black-box attacks.

Detection and Rejection: To mitigate the risk of adversarial examples, detection procedures have been proposed for identifying and rejecting adversarial examples [67; 31; 81; 41; 3]. For instance, [67; 41] stated that adversarial samples are actually statistically different from clean samples. So [41] augments the output of a CNN with an extra dustbin label (a.k.a. reject option), then train it on clean training samples and their corresponding adversarial samples (assigned to dustbin) in order to enable CNNs to detect and reject such adversarial samples. However, this assumes that all adversarial sample generation approaches are known and that a diverse set of adversarial samples are accessible for training. Li [67] utilized a cascade classifier to detect adversarial examples, however [17] demonstrated that this method is not always effective. In [31], authors used kernel density estimation in the feature space to identify adversarial samples, with mixed results given that some of the adversarial samples still become entangled with clean samples, which appears difficult to handle by kernel methods. In [80] one or more autoencoder networks are used for detecting adversarial examples by using the reconstruction error for adversarial examples. In [131], authors proposed detecting adversarial examples by looking at the difference between the prediction of three CNNs trained on the original images, modified images with reduced the color bit depth of pixels, and modified images that are spatially. Further, in addition to learning two or more additional CNNs, they need to learn a threshold for detection over adversarial examples which may change from one dataset to another.

Robust CNNs: In [95], authors reduce the effectiveness of adversarial samples by using distillation networks. Also, in [94], authors reduce the sensitivity of the CNNs to noise by producing very small gradients at the cost of sacrificing accuracy. However, both those methods are not effective for transferable perturbations and further it has been shown that they can be broken easily [16]. There are couple of works which employed a pre-processing step to denoise inputs in order to classify adversarial examples correctly. For example, in [12], authors used PCA to reduce feature space and remove unnecessary information including noise, and in [25] authors used JPEG compression to remove noise from images and then classify the compressed images. However, both reduce the the accuracy of the CNN and are therefore not practical on large datasets.

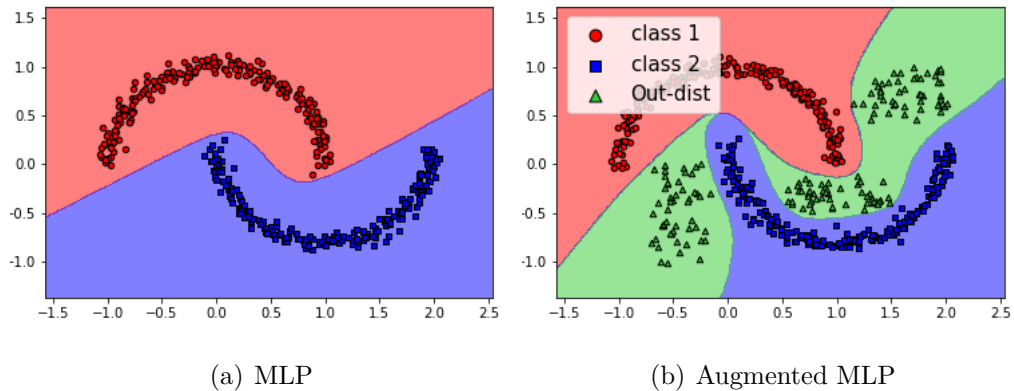


Figure 5.2: Two-moons synthetic dataset. (a) MLP trained on only in-distribution samples (b) the augmented MLP trained on both in-distribution and some out-distribution samples

5.3 Augmented Convolutional Neural Network

It has been argued that a central element explaining the success of deep neural networks is their capacity to learn distributed representations [9]. Indeed, this allows such models to perform well in the regions that are only sparsely sampled in the training set, especially in a very high dimension space. However, neural networks make totally arbitrary decisions in the regions that are outside of the distribution of the learned concepts, leading to over-generalization. For instance, the classification regions achieved from two MLPs on the well-known two-moons dataset are illustrated in Fig. 5.2. Adding a *dustbin* class by including some out of distribution samples to the training set leads to more accurate decision regions, and therefore the over-generalization effect in these out-distribution regions can be reduced. Here, we leverage this to reduce over-generalization by training an augmented CNN with a *dustbin* class for which available natural out-distribution images are used as training samples. We then investigate the effect of this over-generalization reduction on five known powerful adversarial attacks (discussed in Chapter 2). Indeed, as seen in Figure 5.3, we find that the over-generalization reduction leads to a more expressive feature space where all natural out-distribution samples along with many black-box adversarial examples are separated from in-distribution samples to be classified as belonging to the *dustbin* class. Further, some adversarial instances are even placed very close to their corresponding true class, leading the augmented CNNs to classify them correctly.

Interpolated Data: To increase the rejection power of the augmented CNN on

adversarial examples generated using powerful attack algorithms that generate a small amount of smart adversarial perturbations (e.g. DeepFool and C&W attacks), we utilize some interpolated images that are generated from in-distribution samples as training sample for *dustbin* class. Our intuition is that an adversarial example simultaneously contains the relevant features from two classes (*i.e.*, the target class and the true class), where the features of the fooling class are hardly visible while the features related to the true class can be recognizable for human observers. Accordingly, we interpolate some images from in-distribution samples and ensure each resulting image has the features from two different classes. To create interpolated data, we simply add two weighted images together as follows:

$$I_{c_i, c_j, \alpha} = \alpha I_{c_i} + (1 - \alpha) I_{c_j} \quad (5.1)$$

Here, I_{c_i} and I_{c_j} are images from source and target classes. Also, α is a mixing-ratio parameter.

Pair Selection: For generating interpolated data, considering all possible combinations of classes for all training samples of a large-scale dataset and using them for training can be computationally expensive. Therefore, we randomly select a subset of samples from each source class (choosing 1500 samples out of 5000 samples in our simulation). Then, for each selected source sample, its nearest neighbor which is located in another class is found. To do so, we exploit the penultimate layer of a naive CNN as a feature extractor [10; 9] so that we can accurately measure the similarity between images in the feature space rather than in the pixel space.

Furthermore, finding nearest neighbors in feature space is considerably faster than finding them in pixel space due to noticeably smaller dimensionality of feature space (*e.g.*, 512 vs. 3072 for CIFAR).

Finally, using $\alpha = 0.5$, we generate an interpolated sample for each pair of the source image and its target image. Therefore, we can say that interpolated image samples belong equally to the source and target classes. Larger (or smaller) values for α leads the accuracy of CNNs to drop and false positive rate to increase, since interpolated data would be very similar to clean samples.

Out-Distribution Samples: To choose out-distribution samples, we considered images with objects that are totally different from objects in training and test set images. The number of out-distribution samples added to the dustbin class depends on the number of clean examples and the number of classes. A very large dustbin class compared to other classes leads the generalization error to increase and a small one does not improve the resiliency of our augmented CNN. For our evaluation, we increased the training set by 50%.

5.4 Evaluation

Using MNIST [63] and CIFAR-10 [55], we empirically evaluated our proposed method against black-box adversarial samples which are generated using five different well-known attack algorithms. Specifically, we compared the error rates of augmented CNNs with that of naive CNNs for adversarial examples. Here the error rate represents the percentage of wrong decisions made by CNNs for adversarial ex-

amples. That is, where the adversarial examples are neither rejected nor correctly classified. Therefore, we compute the error rate as $1 - (\text{accuracy} + \text{rejection rate})$. We also compared the accuracy of the naive and augmented CNNs for clean samples.

MNIST with NotMNIST: MNIST is a popular machine learning dataset that contains labeled gray scale images, where each image holds a hand-written digit.

As out-distribution samples for MNIST, we considered NotMNIST dataset² that consists of 18,724 letters A-J printed with different font styles. Images of both MNIST and NotMNIST datasets have the same size images (28×28 pixels). We used a CNN named `cuda-convnet` that has three convolution layers with 32, 32, and 64 filters of 5×5 , respectively, and one Fully Connected (FC) layer with softmax activation function³. In addition, dropout with $p = 0.5$ is used at the FC layer for regularization. To train an augmented version of `cuda-convnet`, we utilized a training set comprising 50K MNIST training samples as in-distribution data and 10K randomly selected samples from NotMNIST dataset along with 15K interpolated samples (see Section 5.3) generated from MNIST training samples as out-distribution samples. The remaining samples from NotMNIST ($\approx 8K$) in conjugation with MNIST test samples are considered for evaluating the augmented CNN.

CIFAR-10 with CIFAR-100: Training and test sets of CIFAR-10 contain 50K and 10K RGB images (32×32 pixels each).

²Available at <http://yaroslavvb.blogspot.ca/2011/09/notmnist-dataset.html>.

³To read more about the configuration of this CNN, the readers should refer to <https://github.com/dnouri/cuda-convnet/blob/master/example-layers/layers-18pct.cfg>

As out-distribution samples for CIFAR-10, we consider CIFAR-100 dataset. To reduce a conceptual overlap between the labels from CIFAR-10 and CIFAR-100, we ignore super-classes of CIFAR-100 that are conceptually similar to CIFAR-10 classes. So, vehicle 1, vehicle 2, medium-sized mammals, small mammals, and large carnivores are excluded from CIFAR-100. Note, all the images are scaled to $[0, 1]$, then normalized by mean subtraction over the CIFAR-10 training set. For CIFAR-10, we chose VGG-16 [110] architecture that has 13 convolution layers with filter size 3x3 and three FC layers. To train an augmented VGG-16, 15K randomly selected samples from the non-overlapped version of CIFAR-100 along with 15k interpolated samples from CIFAR-10 training set (labeled as dustbin class) are appended to CIFAR-10 training set.

Evaluation: As adversarial examples are transferable to other CNNs [121; 93], we learned adversarial examples using T-FGS, FGS and DeepFool approaches over independently trained instances of cuda-convnet CNN for MNIST and VGG16 for CIFAR-10. All correctly classified test samples from each dataset are regarded for adversarial example generation by each of the aforementioned attack algorithms. For FGS and T-FGS attacks, we utilized $\epsilon = 0.2$ for MNIST and $\epsilon = 0.03$ for CIFAR-10. For I-FGS attacks, $\epsilon = 0.003$ along with $\alpha = 0.05$ for CIFAR-10 and $\epsilon = 0.02$ along with $\alpha = 0.2$ for MNIST are used. To generate targeted Carlini attack (called C&W) [18], we used the authors' github code. Due to large time complexity of C&W, we considered 100 randomly selected images for each dataset. For each selected image, as was done in previous work [131], two targeted adversarial samples are generated, where the target classes are the least likely

and most likely classes according to the predictions provided by the underlying CNN. Thus, in total 200 C&W adversarial examples are generated per dataset. To increase transferability of C&W, we utilized $\kappa = 20$ for MNIST and $\kappa = 10$ for CIFAR-10 (see Fig.5.1 for some adversarial examples).

Training set: <in-dist, out-dist.>		<MNIST, ->	<MNIST, NotMNIST>	<MNIST, NotMNIST+intrpl.>	<CIFAR-10, ->	<CIFAR-10, CIFAR100>	<CIFAR-10, CIFAR100+intrpl.>
Model		Naive CNN	Augmented CNN	Augmented CNN	Naive VGG	Augmented VGG	Augmented VGG
In-dist. test	Acc.	99.5	99.47	99.48	90.53	88.58	86.65
	Rej.	-	99.96	99.98	-	95.36	96.21
Out-dist. test	Acc.	35.14	19.15	0.35	36.16	27.65	23.94
	Rej.	-	65.19	99.59	-	38.94	49.23
	Err.	64.86	15.66	0.06	63.84	33.41	26.83
	Acc.	16.37	30.97	0.0	50.34	45.98	41.92
	Rej.	-	27.08	100	-	18.57	25.88
	Err.	83.63	41.95	0.0	49.66	35.45	32.2
	Acc.	19.99	1.17	0.0	36.24	27.06	24.2
	Rej.	-	95.92	100	-	40.54	50.77
	Err.	80.01	2.91	0.0	63.76	32.4	25.03
	Acc.	1.89	11.45	5.37	56.82	45.63	42.31
	Rej.	-	4.72	89.84	-	31.0	38.86
	Err.	98.11	83.83	4.8	43.18	23.37	18.83
	Acc.	22.49	27.5	7.5	42.5	46.5	39
	Rej.	-	5.99	77.49	-	18.5	39.5
	Err.	77.51	66.51	15.01	57.5	35	21.5
Average Error rate		80.82	42.17	3.97	55.59	31.92	24.88

Table 5.1: Performance on black-box adversaries attacks. “Acc.” corresponds to accuracy (the rate of correctly classified samples), “Rej.” is the rejection rate, while “Err.” is the misclassification rate All results reported are percentages (%).

The results are shown in Table 5.1, where, "Acc." denotes the accuracy of models in classification of adversarial examples, and column “Rej.” represents the percentage of adversarial samples rejected by the augmented CNNs (*i.e.*, classified as *dustbin*). Finally, "Err" ($= 1 - ("Acc." + "Rej.")$) shows the misclassification rate (*i.e.* the percentage of wrong decisions). As seen in Table 5.1, augmented CNN maintains the same accuracy as naive CNN for clean MNIST test samples, while accuracy of augmented VGG on clean CIFAR-10 test samples drops by $\approx 4\%$ points when compared to naive VGG.

Even though our augmented CNNs are not trained on adversarial inputs generated using any specific algorithm, they were able to reject (classify as *dustbin*)

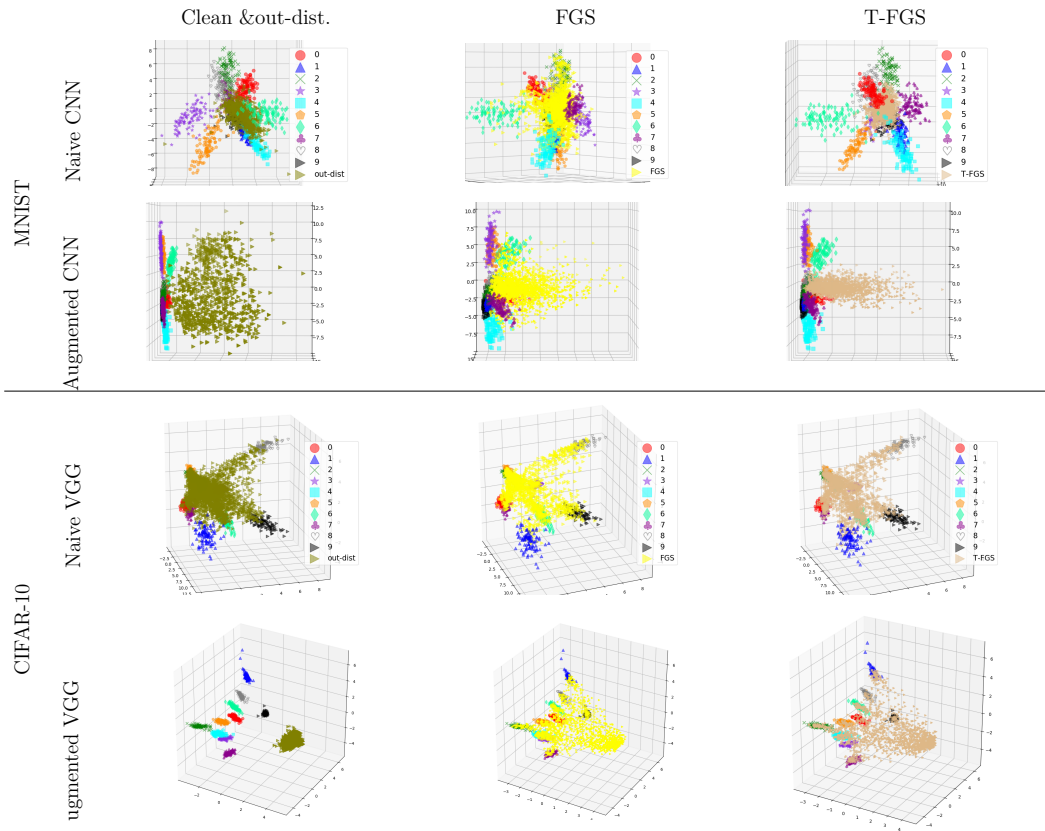


Figure 5.3: Visualization of some randomly selected test samples and their corresponding adversaries (FGS and T-FGS) in the feature spaces (the penultimate layer) learned by a naive CNN and an augmented CNN. To produce and manipulate the 3D plots refer to https://github.com/mahdaneh/Out-distribution-learning_F_Svisualization

most of the adversarial inputs generated using 5 well-known attack algorithms proposed in literature, and even correctly classify a small portion of them. For example, our augmented CNN either rejected or correctly classified almost 96% of the adversarial examples on average, while the Naive CNN misclassified 79% of them. Similarly, for CIFAR-10, the misclassification rate reduces to 24% for augmented VGG when the Naive VGG misclassifies 55% of the adversarial sample

on average.

Further, as shown in this table, adding interpolated data to dustbin class improved the adversarial sample detection rate over just using natural out-distribution data.

Compared to the most recent defense approach [131], we achieved a lower error rate on average over FGS, I-FGS and DeepFool attacks against CIFAR-10 and MNIST. For FGS attacks against MNIST, our error rate was 1% more than what was reported in [131] (0%). However, unlike the approach in [131] we achieved the low error rate (1%) without having to learn additional CNNs. For C&W attacks though, our method had higher error rate (15.1% for MNIST and 21.5% for CIFAR-10) compared to what was reported in [131] (0%). However, apart from not having to learn additional CNNs, we also used a higher⁴ value for κ ($\kappa = 20$ for MNIST; $\kappa = 10$ for CIFAR-10) to generate more transferable attacks. For example, for $\kappa = 0$ our naive MNIST CNN itself was able to correctly classify 97% of C&W adversarial samples as compared to only 22.49% of C&W adversarial samples for $\kappa = 20$.

Moreover, to show how adding a *dustbin* class can effect the feature space learned by a CNN, we plotted the feature spaces obtained from the last convolutional layers of a naive CNN and its corresponding augmented CNN for clean test samples and their corresponding black-box FGS and T-FGS adversarial examples.

In Figure 5.3, one can see that adversarial examples are located very close to the

⁴This is assuming that the authors of [131] used κ value of 0 for C&W attacks as they did not explicitly report the κ value other than saying they used the original implementation of C&W which sets $\kappa = 0$ by default. A lower value of κ leads to lower transferability.

in-distribution samples in the feature space of a Naive CNN. While our augmented CNN, which is trained on both in-distribution and natural out-distribution samples, is able to disentangle the adversarial samples from in-distribution samples in its feature space even though it was not trained on adversarial examples generated using any specific attack approach.

5.5 Adversarial Profile for Out-Distribution Samples Detection

Similar to ODIN [68], we proposed to employ an adversarial attack itself in service of detecting out-distribution and adversarial samples, but instead of training an adversarial example for each input, we trained a set of adversarial perturbations for each class. In other words, we proposed to create a set of such targeted adversarial perturbations for each source class, where each perturbation is designed to move an instance of the source class to a different target class. We refer to this set of adversarial perturbations as an *adversarial profile* of the source class. Let assume that the given in-distribution samples belong to c classes. For a given CNN, adversarial profile of i^{th} class (C_i) is a set of adversarial perturbations $\{\delta_{i,1}, \dots, \delta_{i,i-1}, \delta_{i,i+1}, \dots, \delta_{i,c}\}$ that satisfy the following two properties: i) adding $\delta_{i,j}$ to any clean sample from class i leads the target CNN to misclassify that sample to class j (i.e., if $x \in c_i$, $\operatorname{argmax} F(x + \delta_{i,j}) = j$) with high probability; and ii) adding $\delta_{i,j}$ to any clean sample from other classes ($\neq i$), would lead the CNN to misclassify that sample to any other class except j (i.e., if $x \notin c_i$, $\operatorname{argmax} F(x + \delta_{i,j}) \neq j$).

How to Learn Adversarial Profile: Finding an adversarial perturbation $\delta_{i,j}$ that is

be able to fool all samples from class i to target class j is hard and computationally expensive. Therefore, we only use n randomly selected samples from each source class to learn an adversarial perturbation and accept it for use in the adversarial profile if it can fool the CNN for at least $p * n$ of them ($0 < p < 1$).

We extended C&W [18] attack model to generate an adversarial perturbation that is able to work for a batch of images and get them misclassified to a targeted class as below.

$$\delta_{i,j} = \min_{\delta} \sum_{x_t \in X} \max(\max_{k, k \neq j} \{F(x_{pert.,t})_k\} - F(x_{pert.,t})_j, -\kappa) \quad (5.2)$$

where X is set of n inputs belonging to class i ($\{x_t \in C_i | t = 1 \dots n\}$) used for learning adversarial profiles.

How to Use Adversarial Profile to Detect: To detect out-distribution and adversarial examples, we evaluate the behaviour of input data after applying an adversarial profile. All perturbations in an adversarial profile do not have same effect. To assess a perturbation's effectiveness we define two metrics, *Intra-Class Transferability* and *Inter-Class Transferability*.

- *intra-class transferability*($p_{i,j}$): The probability of fooling CNN to the target class j for samples from source class i is $p_{i,j}$ (i.e., $p(\operatorname{argmax} F(x + \delta_{i,j}) = j | x \in C_i) = p_{i,j}$).
- *inter-class-transferability*($e_{i,j}$): The probability of fooling the CNN to the target class j for samples from other classes ($\neq i$) is $e_{i,j}$ (i.e., $p(\operatorname{argmax} F(x +$

$$\delta_{i,j}) == j | x \notin C_i) = e_{i,j}.$$

To estimate intra-class transferability ($p_{i,j}$) of a perturbation $\delta_{i,j}$, we sample from source class i and apply the perturbation $\delta_{i,j}$ on the samples and measure how many of them are misclassified to the target class j . To estimate inter-class transferability ($e_{i,j}$), we sample from other classes ($\neq i$) and apply adversarial profile $\delta_{i,j}$ on them. Then we measure how many of them are misclassified to the target class j .

$$p_{i,j} = \frac{\sum_i^n I(\operatorname{argmax} F(x + \delta_{i,j}) == j)}{n}, \quad e_{i,j} = \frac{\sum_i^n I(\operatorname{argmax} F(x + \delta_{i,j}) == j)}{n} \quad (5.3)$$

Out-distribution samples do not belong to in-distribution space and therefore they do not have source or target class features. Hence applying an adversarial perturbation designed for in-distribution samples to out-distribution samples should induce a different behavior than expected. To detect out-distribution samples, we apply the perturbations from the adversarial profile associated with the assigned class on the input sample and compute a score that measures how close its behaviour is to that expected of an in-distribution sample of the assigned class.

Unlike out-distribution samples that may not have any in-distribution class features, adversarial samples tend to have features of both original class (albeit weakened) and fooling class. Hence, when an adversarial profile associated with the fooling class is applied it tends to work well making it hard to distinguish from clean samples. However, unlike with clean class samples, for adversarial samples adversarial profile associated with other classes also tends to work well because of

weakened features due to the perturbation. We leverage this difference to detect adversarial samples. To increase the accuracy, we test every input sample with the adversarial profile of the assigned class, and also using adversarial profiles of K other randomly selected classes.

For a received input classified to class i , we compute its score as following:

$$\text{score} = \frac{1}{\sum_j (p_{i,j} - e_{i,j})} \sum_j (p_{i,j} - e_{i,j}) \times I(\text{argmax } F(x) == j) \quad (5.4)$$

For a clean in-distribution sample, and an ideal adversarial profile this score will be close to 1 as all perturbations will take the sample to the expected target class. For out-distribution samples, with high probability this will be less than 1. We find and use a detection threshold to distinguish between in- and out-distributions samples.

5.5.1 Evaluation

We evaluated our proposed method on MNIST [63] dataset. We learned a CNN that reached to 99.2% accuracy on test data. We sampled 100 images for each class from test data to create adversarial profiles with p value of 0.9 which means adversarial profiles can successfully fool at least 90% of samples used for training to the target class. To assess learned adversarial examples, we randomly took $n = 100$ samples from each class and applied its adversarial profile on them. we estimate the intra-class transferability ($p_{i,j}$) and inter-class transferability ($e_{i,j}$).

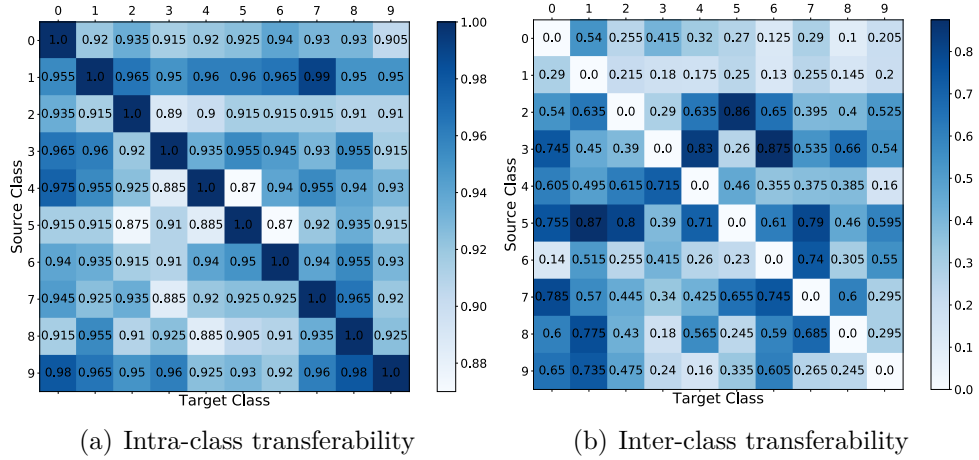


Figure 5.4: Intra and Inter class transferability matrices. The element at $[i, j]$ in Inter class transferability matrix represents the value of $p_{i,j}$. Similarly, the element at $[i, j]$ in intra class transferability matrix represents the value of $e_{i,j}$. The larger value for $p_{i,j}$ and lower value for $e_{i,j}$ are preferred.

The estimated $p_{i,j}$ and $e_{i,j}$ values for MNIST dataset are shown in Figure 5.4. For example, adversarial perturbation $\delta_{5,4}$ has intra-class transferability value of 0.885 which means that this adversarial profile can cause 88.5% of the samples of class 5 to be misclassified to class 4. Moreover this adversarial perturbation causes samples from other classes to misclassify to target class 4 with probability of 71%. Therefore, this adversarial perturbation is not ideal and will have less weight ($0.885 - 0.71 = 0.175$) in our decision-making process. On the other hand, $\delta_{0,8}$ is a good perturbation because it has a high intra-class transferability of 93% and a low inter-class transferability of 10% and will be weighted more ($0.93 - 0.10 = 0.83$). To estimate the best threshold τ , we minimized false positive rate and maximized

Table 5.2: Adversarial and out-distribution detection rates.

Input	Detection Rate	Parameters
SVHN	97%	-
ImageNet	92%	-
CIFAR-100	92%	-
LSUN	97%	-
FGS [36]	63%	$\epsilon = 0.2$
T-FGS [36]	65%	$\epsilon = 0.2$
DeepFool [82]	59%	
I-FGS [57]	64%	$\epsilon = 0.02, \alpha = 0.2$ # of iter. = 20

detection rate on in-distribution samples as follow:

$$\tau = \max_{\tau} \sum_i \frac{1}{2} [I(\text{score}(x_i + \delta_{i,j} | x_i \in C_i) > \tau) + I(\text{score}(x_i + \delta_{i,j} | x_i \notin C_i) < \tau)] \quad (5.5)$$

To estimate $\text{score}(x_i + \delta_{i,j} | x_i \in C_i) > \tau$, we took 100 samples from different classes and then applied out-distribution method detection with their true label. Similarly, to estimate and $\text{score}(x_i + \delta_{i,j} | x_i \notin C_i) < \tau$, we used the same samples with labels different from true labels, therefore adversarial profiles of the fake labels will be applied on them (to satisfy condition of $x_i \notin C_i$).

We considered the 4 datasets, SVHN [86], ImageNet [56], LSUN [134] and CIFAR-100 [55] as out-distribution samples to evaluate our method. We sampled 100 images from each dataset and applied our detection algorithm on them. As shown in Table 5.2, our proposed method is able to reject at least 92% of out-distribution samples from ImageNet and CIFAR-100, and as high as 97% from SVHN and LSUN. Similar to out-distribution samples, we used 4 different attack models (FGS,T-FGS,DeepFool,I-FGS) to generate $n = 100$ adversarial test

examples. As shown in Table 5.2, our method can detect between 59% – 65% of adversarial examples (DeepFool and T-FGS respectively). Compared to previously proposed methods (e.g. [4; 131]), our approach has a lower detection rate. But in contrast to previous work, we do not need to retrain the CNN, and we do not need to have access to a lot of out-distribution or adversarial samples. The only thing we need is a small in-distribution set to make a given CNN robust against out-distribution samples and majority of adversarial samples.

5.6 Future Work

Here we discuss two interesting future directions for developing robust CNNs.

Learning a Synthetic Protective Out-distribution Set: Here, we showed that adding a small set of out-distribution set along with interpolated images can make CNNs significantly robust against adversarial perturbations. In [5], it has been shown that the selection of out-distribution set can affect on the performance of augmented CNNs, so they proposed metrics to select the most suitable out-distribution set for a given in-distribution task. However, the main question is if one can create a synthetic out-distribution samples or enrich the existing out-distribution one with synthetic data?

Extending Adversarial Profile for Larger Datasets: In this chapter, we showed that one can learn adversarial profile for a small pre-trained CNN to detect out-distribution and adversarial samples. It would be interesting to explore this method for larger CNNs and improve the false positive rate using a Encoder-Decoder neu-

ral networks. Also, here we used C& W attack [18] for generating adversarial profile. Other approaches for learning these targeted universal perturbation can be investigated for future work.

5.7 Conclusion

The popularity of CNNs in safety and security sensitive applications necessitate the need for the robust and secure CNNs which can detect and reject suspicious inputs. Here, we proposed two different approaches of making CNNs robust, leaning Augmented CNNs on a few out-distribution samples and Adversarial Profile, a post-processing method used a small set of adversarial perturbations. Both this methods are simple and efficient and can detect unseen adversarial examples and out-distribution samples.

Chapter 6: Conclusion

In this dissertation, we proposed two simple and effective adversarial perturbation generation methods as image privacy defense, one learning-based adversarial perturbation method called Universal Ensemble Perturbation (UEP), and one semantic-based adversarial perturbation method called k-Randomized Transparent Image Overlays (k-RTIO). We showed that these two methods satisfy all usability requirements and they are practical for real-world applications. We evaluated these two methods on well-known offline and online classifiers and demonstrated for specific hyper-parameters, our methods can fool CNN classifiers for at least 85% of perturbed images. We also investigated the methods that an adversary may use to reduce the adversarial perturbations and observed that an adversary can use estimation removal method to estimate UEP perturbation and obtain classifiable images for UEP perturbed images. In addition, she can improve her CNN classifier's accuracy on k-RTIO images by using robust CNNs trained on being and k-RTIO perturbed images which is computationally expensive.

We also evaluated the survivability of adversarial perturbations including UEP and adversarial examples through super resolution convolutional neural networks. We observed both down-scaled adversarial perturbation/noise and learned adversarial perturbations/noises for LR images lead SRCNNs to generated HR adversarial images that can fool CNN classifiers. We also demonstrated an imperceptible

adversarial noise learned over only a small local CNN(s) leads the quality of generated HR images by SRCNNs to downgrade significantly.

Finally, the popularity of CNN classifiers in safety sensitive applications necessitates the need for developing robust and secure CNNs. Therefore, we proposed two methods of Augmented CNNs and Adversarial profiles for adversarial examples and out distribution samples detection and rejection. We demonstrated that our Augmented CNNs can improve adversarial examples detection and rejection significantly just by adding a small set of out-distribution samples to training set. Moreover, our adversarial profiles can detect out-distribution samples on a small pre-trained CNN using a set of universal targeted perturbations for each class that only work for samples belonging to this class.

Bibliography

- [1] An app that encrypts your photos from camera to cloud. <https://www.wired.com/story/pixek-app-encrypts-photos-from-camera-to-cloud/>. Accessed: 2019-11-30.
- [2] Planet selfie. <https://www.businessinsider.com/were-now-posting-a-staggering-18-billion-photos-to-social-media-every-day-2014-5>. Accessed: 2019-11-30.
- [3] Mahdieh Abbasi and Christian Gagné. Robustness to adversarial examples through an ensemble of specialists. *arXiv preprint arXiv:1702.06856*, 2017.
- [4] Mahdieh Abbasi, Arezoo Rajabi, Christian Gagné, and Rakesh B Bobba. Towards Dependable Deep Convolutional Neural Networks (CNNs) with Out-distribution Learning. *Workshop on Dependable and Secure Machine Learning*, 2018.
- [5] Mahdieh Abbasi, Changjian Shui, Arezoo Rajabi, Christian Gagné, and R. Bobba. Toward metrics for differentiating out-of-distribution sets. *European Conference on Artificial Intelligence (ECAI)*, 2020.
- [6] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.
- [7] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 2018.
- [8] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.
- [9] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

- [10] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *International Conference on Machine Learning*, pages 552–560, 2013.
- [11] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '00*, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [12] Arjun Nitin Bhagoji, Daniel Cullina, C Sitawarin, and Prateek Mittal. ‘enhancing robustness of machine learning systems via data transformations, 2017.
- [13] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter Belhumeur, and Shree K. Nayar. Face swapping: Automatically replacing faces in photographs. *ACM Trans. Graph.*, 27(3):39:1–39:8, August 2008.
- [14] Christopher Bourez. *Course 2: build deep learning neural networks in 5 days only*, 2018. <http://christopher5106.github.io/deep/learning/2018/10/20/course-two-build-deep-learning-networks.html>.
- [15] J. F. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, Oct 1998.
- [16] Nicholas Carlini and David Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016.
- [17] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.
- [18] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017.
- [19] Felipe Carmo, Joaquim Assis, Vania Estrela, and Alessandra Coelho. Blind signal separation and identification of mixtures of images. pages 337 – 342, 12 2009.

- [20] Varun Chandrasekaran, Chuhan Gao, Brian Tang, Kassem Fawaz, Somesh Jha, and Suman Banerjee. Face-off: Adversarial face obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2:369–390, 2021.
- [21] Zhiyi Cheng, Xiatian Zhu, and Shaogang Gong. Low-resolution face recognition. In *Asian Conference on Computer Vision*, pages 605–621. Springer, 2018.
- [22] Kenta Chinomi, Naoko Nitta, Yoshimichi Ito, and Noboru Babaguchi. Prisure: Privacy protected video surveillance system using adaptive visual abstraction. In *Proceedings of the 14th International Conference on Advances in Multimedia Modeling*, MMM’08, pages 144–154, Berlin, Heidelberg, 2008. Springer-Verlag.
- [23] Jun-Ho Choi, Huan Zhang, Jun-Hyuk Kim, Cho-Jui Hsieh, and Jong-Seok Lee. Evaluating robustness of deep image super-resolution against adversarial attacks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 303–311, 2019.
- [24] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.
- [25] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E Kounavis, and Duen Horng Chau. Keeping the bad guys out: Protecting and vaccinating deep learning with jpeg compression. *arXiv preprint arXiv:1705.02900*, 2017.
- [26] Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Siwei Li, Li Chen, Michael E Kounavis, and Duen Horng Chau. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–204. ACM, 2018.
- [27] Benedikt Driessen and Markus Dürmuth. *Achieving Anonymity against Major Face Recognition Algorithms*, pages 18–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [28] Ivan Evtimov, Pascal Sturmfels, and Tadayoshi Kohno. Foggysight: A scheme for facial lookup privacy. *arXiv preprint arXiv:2012.08588*, 2020.

- [29] Liyue Fan. Image pixelization with differential privacy. In *Data and Applications Security and Privacy XXXII - 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16-18, 2018, Proceedings*, pages 148–162, 2018.
- [30] Liyue Fan. Practical image obfuscation with provable privacy. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME), 2019*, 2019.
- [31] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.
- [32] Ronald Aylmer Fisher, Frank Yates, et al. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.*, (6th ed), 1963.
- [33] Chuhan Gao, Varun Chandrasekaran, Kassem Fawaz, and Somesh Jha. Face-off: Adversarial face obfuscation. *arXiv preprint arXiv:2003.08861*, 2020.
- [34] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. Sirius: Securing remote untrusted storage. In *NDSS*, volume 3, pages 131–145, 2003.
- [35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [36] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.
- [37] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.
- [38] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.
- [39] Ralph Gross, Edoardo Airoldi, Bradley Malin, and Latanya Sweeney. Integrating utility into face de-identification. In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies, PET’05*, pages 227–242, Berlin, Heidelberg, 2006. Springer-Verlag.

- [40] Ralph Gross, Latanya Sweeney, Jeffrey Cohn, Fernando Torre, and Simon Baker. Face de-identification. *Protecting Privacy in Video Surveillance*, pages 129–146, 2009.
- [41] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.
- [42] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- [44] Kashmir Hill. *The Secretive Company That Might End Privacy as We Know It*, 1/182020.
- [45] Steven Hill, Zhimin Zhou, Lawrence Saul, and Hovav Shacham. On the (in)effectiveness of mosaicing and blurring as tools for document redaction. *Proc. Privacy Enhancing Technologies*, 2016(4):403–17, October 2016.
- [46] Steven Hill, Zhimin Zhou, Lawrence Saul, and Hovav Shacham. On the (in)effectiveness of mosaicing and blurring as tools for document redaction. In *Proceedings on Privacy Enhancing Technologies*, volume 2016, pages 403–417. Sciendo, 2016.
- [47] Geoff Hinton, Yoshua Bengio, and Yann LeCun. Deep learning NIPS’2015 tutorial. Available at <https://www.iro.umontreal.ca/bengioy/talks/DL-Tutorial-NIPS2015.pdf>, 2015.
- [48] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.
- [49] Hossein Hosseini, Baicen Xiao, Andrew Clark, and Radha Poovendran. Attacking automatic video analysis algorithms: A case study of google cloud video intelligence API. In Roger A. Hallman, Kurt Rohloff, and Victor I. Chang, editors, *Proceedings of the 2017 on Multimedia Privacy and Security, MPS@CCS 2017, Dallas, TX, USA, October 30, 2017*, pages 21–32. ACM, 2017.

- [50] T Huang, GJTGY Yang, and G Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.
- [51] Seong Joon Oh, Mario Fritz, and Bernt Schiele. Adversarial image perturbation for privacy protection – a game theory perspective. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [52] A. Jourabloo, X. Yin, and X. Liu. Attribute preserved face de-identification. In *2015 International Conference on Biometrics (ICB)*, pages 278–285, May 2015.
- [53] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Fast*, volume 3, pages 29–42, 2003.
- [54] A. C. Kokaram, R. D. Morris, W. J. Fitzgerald, and P. J. W. Rayner. Interpolation of missing data in image sequences. *IEEE Transactions on Image Processing*, 4(11):1509–1519, Nov 1995.
- [55] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.
- [56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [57] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.
- [58] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.
- [59] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *Workshop Track of the International Conference on Learning Representations (ICLR)*, 2017.
- [60] Karen Lander, Vicki Bruce, and Harry Hill. Evaluating the effectiveness of pixelation and blurring on masking the identity of familiar faces. *Applied Cognitive Psychology*, 15(1):101–116, 2001.

- [61] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.
- [62] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.
- [63] Yann LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [64] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *International Conference on Learning Representations (ICLR)*, 2018.
- [65] Chaofeng Li and Alan C Bovik. Content-partitioned structural similarity index for image quality assessment. *Signal Processing: Image Communication*, 25(7):517–526, 2010.
- [66] Tao Li and Lei Lin. Anonymousnet: Natural face de-identification with measurable privacy. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.
- [67] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. *CoRR*, [abs/1612.07767](https://arxiv.org/abs/1612.07767), 7, 2016.
- [68] Shiyu Liang, Yixuan Li, and R Srikant. Principled detection of out-of-distribution examples in neural networks. *International Conference on Learning Representations (ICLR)*, 2018.
- [69] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.
- [70] Y. Lin, S. Wang, Q. Lin, and F. Tang. Face swapping under large pose variations: A 3d model based approach. In *2012 IEEE International Conference on Multimedia and Expo*, pages 333–338, July 2012.
- [71] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

- [72] Cheng Ma, Zhenyu Jiang, Yongming Rao, Jiwen Lu, and Jie Zhou. Deep face super-resolution with iterative collaboration between attentive recovery and landmark estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [73] Cheng Ma, Yongming Rao, Yean Cheng, Ce Chen, Jiwen Lu, and Jie Zhou. Structure-preserving super resolution with gradient guidance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [74] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.
- [75] H. S. Malvar, Li-wei He, and R. Cutler. High-quality linear interpolation for demosaicing of bayer-patterned color images. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–485, 2004.
- [76] Byron Marohn, Charles V Wright, Wu-chi Feng, Mike Rosulek, and Rakesh B Bobba. Approximate thumbnail preserving encryption. 2017.
- [77] Byron Marohn, Charles V. Wright, Wu-chi Feng, Mike Rosulek, and Rakesh B. Bobba. Approximate thumbnail preserving encryption. In *Proceedings of the 2017 on Multimedia Privacy and Security, MPS '17*, pages 33–43, New York, NY, USA, 2017. ACM.
- [78] Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *European Conference on Computer Vision*, pages 720–735. Springer, 2014.
- [79] Richard McPherson, Reza Shokri, and Vitaly Shmatikov. Defeating image obfuscation with deep learning. *arXiv preprint arXiv:1609.00408*, 2016.
- [80] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017.
- [81] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *5th International Conference on Learning Representations (ICLR)*, 2017.

- [82] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, June 2016.
- [83] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1765–1773, 2017.
- [84] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [85] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [86] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [87] Elaine M Newton, Latanya Sweeney, and Bradley Malin. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering*, 17(2):232–243, 2005.
- [88] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE international conference on image processing (ICIP)*, pages 343–347. IEEE, 2014.
- [89] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 1st edition, 2009.
- [90] Seong Joon Oh, Rodrigo Benenson, Mario Fritz, and Bernt Schiele. *Faceless Person Recognition: Privacy Implications in Social Media*, pages 19–35. Springer International Publishing, Cham, 2016.
- [91] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Towards a visual privacy advisor: Understanding and predicting privacy risks in images. In

- IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 3706–3715. IEEE Computer Society, 2017.
- [92] Ajit B Pande and Shashi Prabha. Image blind signal separation algorithm based on fast ica. In *IJCA Proceedings on International Conference on Advances in Communication and Computing Technologies 2012*, number 3, pages 21–24. Citeseer, 2012.
- [93] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.
- [94] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *IEEE European Symposium on Security and Privacy*, 2018.
- [95] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 582–597. IEEE, 2016.
- [96] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.
- [97] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. P3: Toward privacy-preserving photo sharing. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 515–528, Lombard, IL, 2013. USENIX.
- [98] A. Rachel Abraham, A. Kethsy Prabhavathy, and J. Devi Shree. A Survey on Video Inpainting. *International Journal of Computer Applications*, 56(9):43–47, October 2012.
- [99] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

- [100] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.
- [101] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.
- [102] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [103] Michael Ryoo, Brandon Rothrock, Charles Fleming, and Hyun Jong Yang. Privacy-preserving human activity recognition from extreme low resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- [104] Somini Sengupta and Kevin J. O’Brien. Facebook can id faces, but using them grows tricky. *The New York Times*, September 2012.
- [105] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: Protecting personal privacy against unauthorized deep learning models. *arXiv preprint arXiv:2002.08327*, 2020.
- [106] Yash Sharma and Pin-Yu Chen. Attacking the madry defense model with l_1 -based adversarial examples. *arXiv preprint arXiv:1710.10733*, 2017.
- [107] Wenzhe Shi, Jose Caballero, Christian Ledig, Xiahai Zhuang, Wenjia Bai, Kanwal Bhatia, Antonio M Simoes Monteiro de Marvao, Tim Dawes, Declan O’Regan, and Daniel Rueckert. Cardiac image super-resolution with global correspondence using multi-atlas patchmatch. In *International conference on medical image computing and computer-assisted intervention*, pages 9–16. Springer, 2013.
- [108] Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Portrait of a privacy invasion. *Proceedings on Privacy Enhancing Technologies*, 2015(1):41–60, 2015.

- [109] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.
- [110] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [111] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.
- [112] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [113] Z. Stone, T. Zickler, and T. Darrell. Autotagging facebook: Social network context improves photo annotation. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2008.
- [114] Qianru Sun, Ayush Tewari, Weipeng Xu, Mario Fritz, Christian Theobalt, and Bernt Schiele. A hybrid model for identity obfuscation by face replacement. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 570–586, Cham, 2018. Springer International Publishing.
- [115] Wanjie Sun and Zhenzhong Chen. Learned image downscaling for upscaling using content adaptive resampler. *IEEE Transactions on Image Processing*, 29:4027–4040, 2020.
- [116] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3476–3483. IEEE, 2013.
- [117] Z. Sun, L. Meng, and A. Ariyaeeinia. Distinguishable de-identified faces. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 04, pages 1–6, May 2015.
- [118] Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.

- [119] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [120] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.
- [121] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
- [122] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.
- [123] Kimia Tajik, Akshith Gunasekaran, Rhea Dutta, Brandon Ellis, Rakesh B. Bobba, Mike Rosulek, Charles V. Wright, and Wu-chi Feng. Balancing image privacy and usability with thumbnail-preserving encryption. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*, 2019.
- [124] Suriyon Tansuriyavong and Shin-ichi Hanaki. Privacy protection by concealing persons in circumstantial video image. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces, PUI ’01*, pages 1–4, New York, NY, USA, 2001. ACM.
- [125] Matt W Thornton, Peter M Atkinson, and DA Holland. Sub-pixel mapping of rural land cover objects from fine spatial resolution satellite sensor imagery using super-resolution pixel-swapping. *International Journal of Remote Sensing*, 27(3):473–491, 2006.
- [126] Matt Tierney, Ian Spiro, Christoph Bregler, and Lakshminarayanan Subramanian. Cryptagram: Photo privacy for online social media. In *Proceedings of the First ACM Conference on Online Social Networks, COSN ’13*, pages 75–88, New York, NY, USA, 2013. ACM.

- [127] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011.
- [128] Charles V. Wright, Wu-chi Feng, and Feng Liu. Thumbnail-preserving encryption for jpeg. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec '15*, pages 141–146, New York, NY, USA, 2015. ACM.
- [129] Charles V Wright, Wu-chi Feng, and Feng Liu. Thumbnail-preserving encryption for jpeg. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, pages 141–146, 2015.
- [130] Zhenyu Wu, Zhangyang Wang, Zhaowen Wang, and Hailin Jin. Towards privacy-preserving visual recognition via adversarial training: A pilot study. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 627–645, Cham, 2018. Springer International Publishing.
- [131] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *Network and Distributed System Security Symposium*, 2018.
- [132] Y. Yang, P. Bi, and Y. Liu. License plate image super-resolution based on convolutional neural network. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 723–727, 2018.
- [133] Minghao Yin, Yongbing Zhang, Xiu Li, and Shiqi Wang. When deep fool meets deep prior: Adversarial attack on super-resolution network. *MM '18*, page 1930–1938, New York, NY, USA, 2018. Association for Computing Machinery.
- [134] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
- [135] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

- [136] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018.
- [137] Qiang Alex Zhao and John T. Stasko. The awareness-privacy tradeoff in video supported informal awareness: A study of image-filtering based techniques. Technical report, Georgia Tech, <http://hdl.handle.net/1853/3452>, 1998.
- [138] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [139] W. W. W. Zou and P. C. Yuen. Very low resolution face recognition problem. *IEEE Transactions on Image Processing*, 21(1):327–340, 2012.

APPENDICES

Appendix A: UEP Setups

A.1 Neural Networks

We trained small convolutional neural networks on limited faces (10 classes of faces). To minimize the network dependencies, we utilized two different CNNs structures trained on different images sizes.

VGG16:

```
model = Sequential()
model.add(Conv2D(128, (5, 5),
                input_shape=shape_image))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2),
                       strides=(2,2), padding='valid'))
model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2),
                       strides=(2,2), padding='valid'))
model.add(Conv2D(512, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2),
```

```

        strides=(2,2), padding='valid'))
model.add(Conv2D(512, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(512, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2),
        strides=(2,2), padding='valid'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(self.num_classes))

```

FaceScrub:

```

model = Sequential()
model.add(Conv2D(32, (3, 3),
        input_shape=shape_image))
model.add(LeakyReLU(alpha=.01))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, (3, 3)))
model.add(LeakyReLU(alpha=.01))

```



```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(128, (3, 3)))
model.add(LeakyReLU(alpha=.01))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(200))
model.add(LeakyReLU(alpha=.01))
model.add(Dropout(0.5))
model.add(Dense(self.num_classes))
```

Appendix B: k-RTIO Setups and Extra Evaluation

B.1 Overlay Images

We used different images of objects, landscape etc. in our set S (initial set of overlay images). We did not have any constraints on selecting the images except not having human faces in there. Because using an image with faces in there may reduce the recognizability of the images by end users.

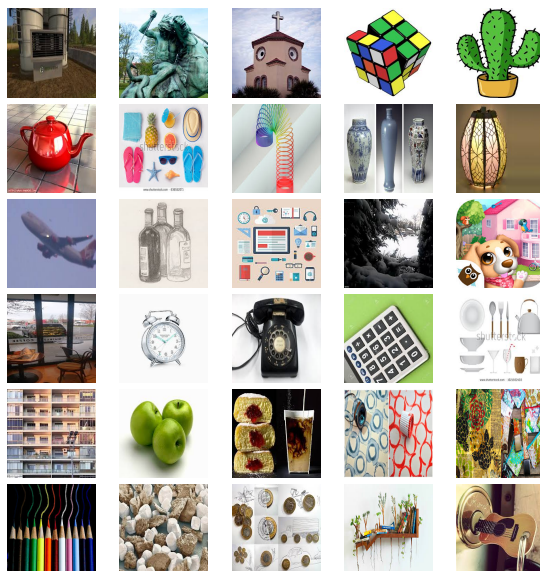


Figure B.1: Set S of the overlay images used in generating k-RTIO images

Set of overlay images used in our experiments are shown in Figure B.1.

B.2 Blind Signal Separation

Blind signal separation (BSS) is used to extract independent signals $X = [x_1^T, \dots, x_n^T]$ which are mixed together by a mixing matrix ($A_{m \times n}$) from the resulting signals $B = A \times X$. The solution would be easy if the adversary knows the mixing matrix ($A_{m \times n}$). However, this matrix is unknown. The adversary can use blind signal separation method to recover the original signal only by considering the correlation between the signals B .

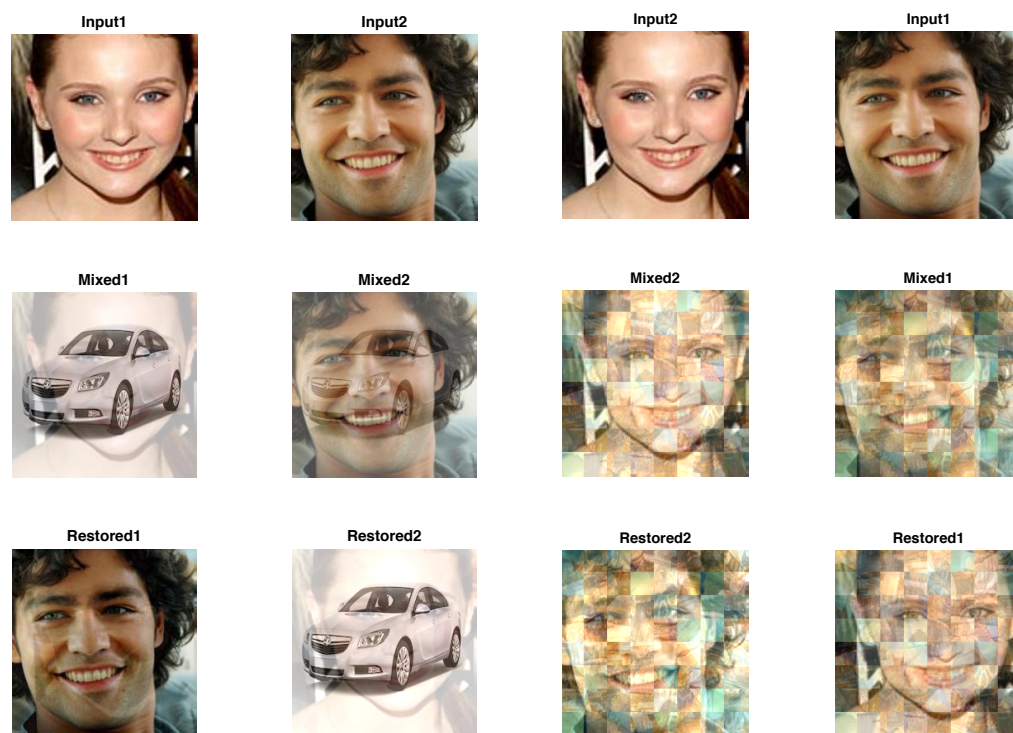
Let, $I = [I_1, \dots, I_m]$ be a set of original images where we want to add the image I' as a new layer with the weight of α ($b_j = \alpha I_j + (1 - \alpha)I'_j$). The user first converts the images to raw data in the form that a signal is represented by as follows:

$$X = \begin{bmatrix} I_{11} & \dots & I_{1n} \\ I_{21} & \dots & I_{2n} \\ \vdots & \vdots & \vdots \\ I_{m1} & \dots & I_{mn} \\ I'_{11} & \dots & I'_{1n} \end{bmatrix}, A = \begin{bmatrix} \alpha & 0 & 0 & \dots & 1 - \alpha \\ 0 & \alpha & 0 & \dots & 1 - \alpha \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \alpha & 1 - \alpha \end{bmatrix} \quad (\text{B.1})$$

and then creates matrix A . The user obtains the perturbed images by multiplying matrix X by A .

The adversary only has access to the mixed signal B and wants to recover the original images. As shown in Figure B.2, we used FastICA [92; 19] method (variant of BSS for images) to recover the I' or the images. As shown in this Figure, the adversary can easily recover the original images from the mixed images. Because

this method initializes the correlation matrix randomly, the final recovered images are different. But it is possible to recover one of the pictures well. It is obvious that just by having one image, we can recover the other images easily. However, in k-RTIO we apply different overlay images for each original image. Further, we also select uniformly at random k overlay images used for any given image from a larger pool of S overlay images which makes finding blocks overlaid with same k blocks difficult. Therefore, BSS is not effective against k-RTIO.



(a) BSS on Transparent Overlay Image Method

(b) BSS on k-RTIO Perturbed Images

Figure B.2: Blind Signal Separation can recover perturbed images by transparent overlays technique. Generating overlay images by permuting the original overlay images' blocks randomly leads BSS to not be able to recover images.