Screen Perturbation

Adversarial Attack and Defense on Under-Screen Camera

Ye, Hanting; Lan, Guohao ; Jia, Jinyuan ; Wang, Qing

DOI
10.1145/3570361.3613278

Publication date
2023

Document Version
Final published version

Published in
ACM MobiCom '23: Proceedings of the 29th Annual International Conference on Mobile Computing and Networking

Screen Perturbation: Adversarial Attack and Defense on Under-Screen Camera

Hanting Ye[†], Guohao Lan[†], Jinyuan Jia[‡], Qing Wang[†]
[†]Delft University of Technology, The Netherlands [‡]The Pennsylvania State University, USA

ABSTRACT

Smartphones are moving towards the *full-screen* design for better user experience. This trend forces front cameras to be placed under screen, leading to *Under-Screen Cameras (USC)*. Accordingly, a small area of the screen is made *translucent* to allow light to reach the USC. In this paper, we utilize the translucent screen's features to inconspicuously modify its pixels, imperceptible to human eyes but inducing perturbations on USC images. These screen perturbations affect deep learning models in image classification and face recognition. They can be employed to protect user privacy, or disrupt the front camera's functionality in the malicious case. We design two methods, one-pixel perturbation and multiple-pixel perturbation, that can add screen perturbations to images captured by USC and successfully fool various deep learning models. Our evaluations, with three commercial full-screen smartphones on testbed datasets and synthesized datasets, show that screen perturbations significantly decrease the average image classification accuracy, dropping from 85% to only 14% for one-pixel perturbation and 5.5% for multiple-pixel perturbation. For face recognition, the average accuracy drops from 91% to merely 1.8% and 0.25%, respectively.

CCS CONCEPTS

• **Security and privacy** → **Usability in security&privacy**;
• **Computing methodologies** → **Adversarial learning**.

KEYWORDS

Under-screen camera, adversarial perturbation, privacy

ACM Reference Format:
Hanting Ye, Guohao Lan, Jinyuan Jia, Qing Wang. 2023. Screen Perturbation: Adversarial Attack and Defense on Under-Screen Camera. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3570361.3613278

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10.
https://doi.org/10.1145/3570361.3613278

1 INTRODUCTION

The pursuit of immersive user experience has encouraged full-screen design in mobile devices. A key innovation enabling full-screen design is the *Under-Screen Camera (USC)* technology, which positions the camera beneath the screen, maintaining the device's sleek aesthetics without compromising functionality. This novel design, which prioritizes an increased screen-to-body ratio, is now prevalent in consumer devices such as smartphones like Samsung Galaxy Fold3/4, ZTE AXON 20/30/40, Xiaomi MIX4, and K50S, and laptops like Thunderobot T-BOOK 14 and Samsung Blade Bezel.

To allow visible light reach the USC and thereby preserve its photographic capabilities [3], a small region of the device's screen is made translucent, leading to a *Translucent Screen Region (TSR)* that positions above the USC. Motivated by this new feature of full-screen consumer devices, in this paper, *we propose the concept of Screen Perturbation, which modifies the pixels displayed on the TSR to nullify deep learning models such as image classification and face recognition models.*

The proposed screen perturbation is a *double-edged sword. On one hand*, it can serve as a defensive mechanism. Like traditional front cameras, hackers can exploit USCs through camfecting [66] to capture images and record videos. With the rise of deep learning models in security-critical applications like face authentication [65], these illegally captured images can be used to train extensive models capable of recognizing millions of individuals without their consent. To tackle this security risk, users can leverage the TSR to embed adversarial perturbations into captured images, thereby protecting themselves from unauthorized deep-learning models. For instance, an unauthorized face recognition system would fail to correctly identify users based on these perturbed images, as depicted in Figure 1. Users can proactively activate specific screen pixels during their interactions with the smartphone, ensuring that any images covertly captured by USC are not correctly identified by unauthorized facial recognition models. Importantly, this does not interfere with other functionalities, such as the motion detection capability of USC. *On the other hand*, the TSR can be exploited by malicious attackers. They can embed adversarial perturbations into captured images to launch adversarial attacks, aiming to disrupt legitimate face recognition systems or fool image classification models. To fully harness the potential of screen perturbation and mitigate its risks, we must address several

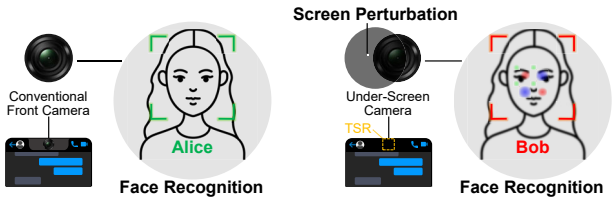


Figure 1: Illustration of screen perturbation in defending unauthorized face recognition: carefully-selected pixels on the translucent screen are lit up, which are 'imperceptible' to human eyes, but could cause an unauthorized system to misrecognize the user Alice as Bob.

unique challenges. For simplicity of presentation, we describe these challenges and our solutions and contributions *from the attacker's perspective* in the rest of this section.

To launch adversarial attacks that can fool applications such as face recognition and image classification, the attacker needs to manipulate the content displayed in the TSR to embed adversarial perturbations in the formed images. This is challenging because the modification in the TSR needs to be carefully designed such that the changes are imperceptible to the users, while the generated image perturbations are powerful enough to fool the deep learning models.

To achieve this goal, the first challenge is to *understand the impact of the TSR's perturbation on the formed image at the USC*. Current studies [67, 74, 75] have discovered that when the translucent screen is not illuminated, it can still scatter and absorb light, leading to a lower signal-to-noise ratio and resulting in a color shift in the formed image; we refer to this as *passive perturbation*. However, the effect of the screen's illuminated pixels on the images captured by the USC has never been explored. Our research reveals that when the TSR is lit for displaying content, the illuminated screen pixels embed various translucent speckled color blocks and color shifts in the formed image. We term this *active perturbation*. Factors such as whether the pixels in the TSR are illuminated, their colors, and their brightness levels strongly affect these perturbations added to the formed image. To address this, we develop a comprehensive model to study the impact of both passive perturbations and active perturbations on USC's image formation. Building upon this model, we successfully create a USC image simulator that generates both passive and active perturbations on captured images simultaneously, allowing us to quantitatively understand the impact of screen perturbation on the under-screen image formation.

The second challenge is to *determine which TSR pixels to manipulate and how, in order to ensure that the created screen perturbation can successfully fool deep learning models*. Although the size of TSR is small (only about 0.1 cm^2), it contains *several thousand to twenty thousand pixels*. For example, the smartphone Fold4, AXON30, and MIX4 have 58∗58, 108∗60, and 108∗60 pixels within their TSR, respectively. Each pixel has multiple R/G/B sub-pixels, making it challenging to identify the most suitable pixels for creating the screen perturbation and how to light them up (using optimal colors and brightness). The simplest way could be to light up all the pixels in the TSR at a proper brightness level to create the screen perturbation. However, this is inefficient and the created screen perturbation will be striking, and therefore, users can perceive it.

To address this challenge, we present the chromaticity destruction and morphology destruction modules. Specifically, the chromaticity destruction determines the optimal color of the screen perturbation and maximizes its attacking region and energy intensity. The morphology destruction module optimizes the position and brightness of the screen-pixel perturbations. Leveraging these two modules, we design a one-pixel perturbation approach, wherein modifying only a single pixel on the screen *(less than 1‰ of the pixels in the translucent screen region)* we can reduce the average image classification accuracy of six deep learning models from 85% to 14%, and reduce the average face recognition accuracy of two deep learning models from 91% to 1.8%. To enhance attack efficiency, we propose a multiple-pixel perturbation that modifies only a few screen pixels *(less than 1% of the pixels in the translucent screen region)* to generate perturbation with a higher adversarial strength, thereby further decreasing the average accuracy of image classification and face recognition to as low as 5.5% and 0.25%, respectively. We summarize our contributions as follows:

- To our best knowledge, we are the first to discover this critical security phenomenon of USC.
- We analyze the imaging formation of USC both theoretically and experimentally. Theoretical models are successfully established, and we build an image captured on the USC simulator based on those models.
- We design and implement one-pixel and multiple-pixel perturbations including chromaticity and morphology destruction modules, which could generate imperceptible but powerful screen perturbations to fool deep learning models.
- We thoroughly evaluate the system on a dataset collected by three smartphones equipped with USC and synthesized datasets generated using our USC image formation simulator.

2 PRIMER ON SCREEN PERTURBATION

2.1 Structure of Translucent Screen Region

In Under-Screen Camera (USC) smartphones [1, 2, 31, 68, 69], the front-facing camera is placed under a small 'translucent' screen area known as the Translucent Screen Region (TSR), which is built on translucent cathode material. As shown in Figure 2(b), the TSR has a different pixel layout from the normal screen region, and can be considered as an RGBG array that consists of multiple *screen-pixel units*. Figure 2(c) shows a typical structure of the screen-pixel unit, which is made up of three R/G/B subpixel sets. Depending on the manufacturer, a R/G/B subpixel set can contain multiple

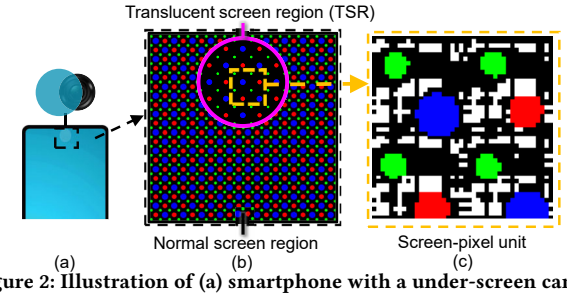


Figure 2: Illustration of (a) smartphone with a under-screen camera; (b) micrograph of the translucent screen region and the normal screen region; and (c) a typical structure of the screen-pixel unit.

subpixels of the same color. For example, in the design shown in Figure 2(c), the R/G/B subpixel set is made up of two red subpixels, four green subpixels, and two blue subpixels.

We can further divide the TSR into three functional areas:

- **Screen-non-pixel area** (the white area) allows visible light to pass through the screen.
- **Control-circuit area** (the black area) is mainly used to control the circuit wiring of the screen, and has a lower light transmission rate than the screen-non-pixel area.
- **Screen-pixel area** (the RGB color matrix) consists of R/G/B subpixel sets that can be lit up to display contents.

2.2 Under-Screen Image Perturbations

The unique structure of the TSR introduces two types of perturbation during image formation. First, as the TSR is placed above the camera, the screen-non-pixel and the control-circuit areas introduce **passive perturbations** on the formed images. As shown in Figure 3(b), passive perturbations are embedded into the image even when the TSR is inactive and the screen is completely turned off due to light scattering and absorption from the fine pixel pitch [75]. The passive perturbation causes a lower signal-to-noise ratio and color shifts in the image [67, 74, 75]. In addition, as shown in Figure 3(c), when the screen is lit up for display, the screen-pixel area introduces **active perturbations** to the image. The lighting of different R/G/B subpixel sets embeds various translucent speckled color blocks and color shifts in the final image.

3 SYSTEM OVERVIEW

3.1 Threat Model

In this work, we investigate and demonstrate that the new type of screen perturbation is a double-edged sword for users. On one hand, malicious attackers can exploit it to embed adversarial perturbations into captured images, launching adversarial attacks aimed at disrupting the performance of legitimate face recognition systems [54] or fooling image classification models [28]. On the other hand, benign defenders, e.g., users, can leverage screen perturbation to protect themselves from unauthorized deep learning models. For instance, users can proactively activate specific screen pixels in use. Thus, any images covertly captured by the under-screen

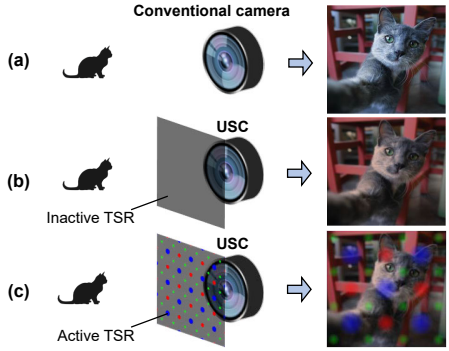


Figure 3: Images formed in different scenarios: (a) pristine image is captured by a traditional front-facing camera; (b) the TSR is inactive but still introduces passive screen perturbation; (c) the TSR is active and introduces both passive and active screen perturbations.

camera would not be correctly identified by unauthorized facial recognition models [53, 65], while other functionalities, such as the motion detection capability of the under-screen camera, would remain unaffected. Next, we motivate our system design from the perspective of the attacker/defender's goal, capability, and background knowledge.

Attacker/defender's goal. We consider an attacker/defender aims to manipulate the content displayed in the TSR to generate and embed adversarial perturbations to images captured by the under-screen camera. As a result, a machine learning model (referred as **target model**) trained on images captured by a standard camera should exhibit low accuracy for those corrupted images. The alterations to the displayed content should be imperceptible and should not interfere with the normal content displayed on the device screen.

Attacker/defender's capability and background knowledge. We begin by assuming that a benign or malicious application is installed by the attacker or defender. This application has the ability to control all screen-pixel units in the TSR and can modify the content displayed on the screen. Android platforms do not restrict UI settings, allowing any applications to set their own UI displays and perturb the camera input. For devices such as Samsung Galaxy Z Fold 4, ZTE AXON 30, and Xiaomi MIX 4 which we investigate in this work, the content displayed on the TSR can be easily modified using the Android Canvas Drawing API [12]. We also assume that these screen-pixel modifications are not confined to a single application but can be implemented across different applications, as Android systems support UI modifications between various applications, including multi-window settings [13] and notification pop-ups [14]. Next, we assume that the attacker or defender has access to an uncorrupted image for obtaining object information. This image can be sourced from various places, such as a social media selfie, and doesn't need to be taken in run-time. For benign applications, users can take an uncorrupted image themselves. For malicious applications, an attacker could use a victim's social media selfie as an uncorrupted image.

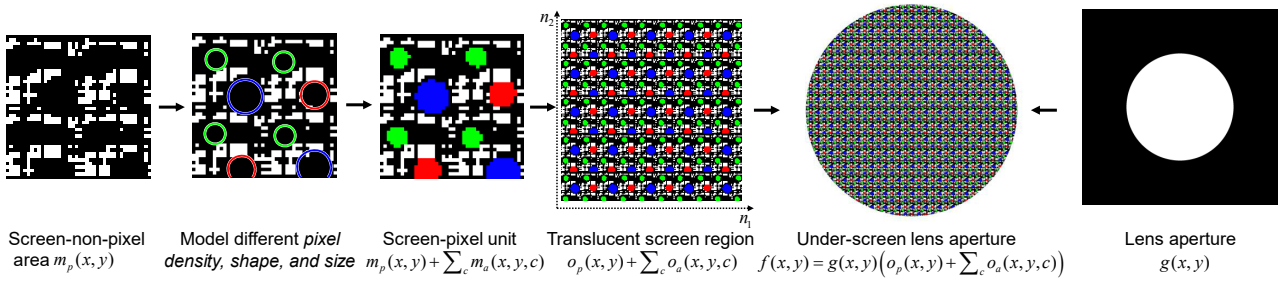


Figure 4: Modeling the effective under-screen lens aperture.

Lastly, the attacker/defender can generate different screen perturbations by modifying the content displayed on the screen for various usage scenarios. These modified pixels on the screen are invisible to the naked eye, ensuring that they do not affect the user's normal use of the smartphone in benign applications and do not alert the user to the embedded screen perturbation in malicious applications. To consider a more realistic scenario, we assume the attacker/defender lacks knowledge about the target model's implementation details used to classify captured images, but they have access to a **surrogate model** for the task, which could have different architecture and hyper-parameters from target models. For simplicity, we describe our design **from the attacker's perspective** in the rest of the paper.

3.2 Overall Design

We first establish image formation models as our design's theoretical basis. We model the active perturbation from the screen-pixel area (in Section 4.2) and simulate how color, brightness, and position of the subpixel set affect the active perturbation. We also model the passive perturbation from the screen-non-pixel area (in Section 4.3). Our under-screen image formation pipeline synthesizes images affected by both perturbations. Building on our theoretical models, we devise mechanisms to configure the TSR for subtle yet effective perturbations, including the chromaticity destruction module (in Section 5.2) and the morphology destruction module (in Section 5.3). To enhance the attack efficiency, we design the multiple-pixel perturbation (in Section 5.4).

4 MODELING USC'S IMAGE FORMATION

We first introduce the concept of blur kernel, then specialize it to model the active and passive screen perturbations.

4.1 Blur Kernel

Light diffraction through the TSR in a camera system can cause image degradation due to the comparable size of the screen-non-pixel area and the visible light wavelength [75]. Following Fourier optics principles [21], we model this diffraction phenomenon using the blur-kernel, which is also known as the point spread function [67]. The blur kernel mathematically depicts the spread of light around a point light source in an image, characterizing the blurring effect of an imaging system or physical medium on the image. The blur kernel is the squared magnitude of the scaled Fourier transform of the under-screen aperture function $f(x,y)$, derived from the product of the camera lens aperture and the TSR:

$$f(x,y) = g(x,y)o(x,y), \tag{1}$$

where (x,y) represents the coordinates of the screen-pixel unit. The camera lens is approximated as a thin lens with an aperture function $g(x,y)$, while the TSR is modeled as a function $o(x,y)$ that maps the light transmission properties of coordinates (x,y) in the TSR to a range between 0 and 1. A value of 1 indicates that light can fully pass through the TSR, whereas a value of 0 indicates that light cannot pass through the TSR. The blur kernel is defined as follows:

$$k(\lambda, F) = \left| \frac{1}{\lambda r_0} F\left(\frac{x}{\lambda r_0}, \frac{y}{\lambda r_0} \right) \right|^2, \tag{2}$$

where λ is the wavelength of light, r_0 is the focal length of the camera lens, and $F(u,v)$ is the Fourier transform of the under-screen aperture function $f(x,y)$. The (u,v) is the mapping of coordinates (x,y) in the frequency domain.

Based on the blur-kernel model, obtaining the under-screen aperture function of active and passive perturbations for TSR is necessary to generate the corresponding blur kernels. The under-screen aperture functions for both types of perturbations will be presented below, as illustrated in Figure 4.

4.2 Modeling Screen's Active Perturbation

USCs in mobile devices like smartphones are placed near the screen for a slim profile, leading to the TSR co-locating with the thin lens aperture. This proximity allows the R/G/B subpixel sets on the TSR to be approximated as point light sources per the Huygens-Fresnel principle [21]. The TSR's screen-pixel area is periodic, with identical screen-pixel units, enabling modulation of the entire area via a single unit. Let D μm be the inter-pixel distance of the pixels in the TSR. D actually determines the screen resolution as $25,400/D$ pixels per inch (= 25,400 μm). The light intensity of screen-pixel area within a screen-pixel unit is denoted as $m_a(x,y,c)$, where $c \in \{R,G,B\}$ represents the R/G/B subpixel sets. This $m_a(x,y,c)$ repeats at a periodicity of D along both axes to form all screen-pixel areas on the TSR. Here, we can set R/G/B subpixel sets *shape*, *size* and *brightness* in the screen-pixel unit to impact the active screen-pixel perturbation.

However, the shape and size of each R/G/B subpixel set are typically predetermined during manufacturing. The active perturbation can still be manipulated by selectively lighting up different R/G/B subpixel sets on the TSR to form different patterns. Using n_1 and n_2 to represent the horizontal and vertical indices in the spatial domain and N_1 and N_2 for the frequency domain, we can express the light intensity of (x, y) of different R/G/B subpixel sets on the TSR $o_a(x, y, c)$ as:

$$o_a(x, y, c) = m_a(x, y, c) * \sum_{n_1} \sum_{n_2} \delta(x - n_1 D)\delta(y - n_2 D), \quad (3)$$

where $\delta(\cdot)$ is the Dirac delta function, also known as the unit impulse [21]. Multiplication in the space domain leads to convolution in the frequency domain, which allows us to obtain $F_a(u, v, c)$, the Fourier transform of the effective aperture at different wavelengths (R/G/B) for active perturbation:

$$F_a(u, v, c) = \sum_{N_1} \sum_{N_2} M_a\left(\frac{N_1}{D}, \frac{N_2}{D}, c\right) G\left(u - \frac{N_1}{D}, v - \frac{N_2}{D}\right), \quad (4)$$

where $M_a(u, v, c)$ represents the Fourier transform of R/G/B subpixel set in the screen-pixel area $m_a(x, y, c)$, and $G(u, v)$ is the Fourier transform of the camera lens aperture $g(x, y)$.

4.3 Modeling Screen's Passive Perturbation

We present a specialized under-screen aperture function expression, focusing on passive perturbation from the screen-non-pixel area. Given its higher light transmittance compared to the control circuit and screen-pixel areas, we can neglect the latter [67]. The screen-non-pixel area within a screen-pixel unit, denoted as $m_p(x, y)$ (the white area in Figure 2), also exhibits periodicity. This allows us to mathematically express the light transmission properties of (x, y) on the TSR's screen-non-pixel area, $o_p(x, y)$, and the Fourier transform of the effective aperture for passive perturbation:

$$o_p(x, y) = m_p(x, y) * \sum_{n_1} \sum_{n_2} \delta(x - n_1 D)\delta(y - n_2 D), \quad (5)$$

$$F_p(u, v) = \sum_{N_1} \sum_{N_2} M_p\left(\frac{N_1}{D}, \frac{N_2}{D}\right) G\left(u - \frac{N_1}{D}, v - \frac{N_2}{D}\right). \quad (6)$$

4.4 Under-Screen Image Formation Pipeline

Based on the analysis in the previous sections, and given a calibrated screen-pixel unit as shown in Figure 4, we can model the degraded images from a scene with both passive and active perturbation of the TSR. For the object-of-interest in the scene $\boldsymbol{I}$, the degraded observation $\boldsymbol{I} \oplus \boldsymbol{z}$ formed on the sensor can be modeled as a convolution process, given by:

$$\boldsymbol{I} \oplus \boldsymbol{z} = \underbrace{(\gamma \hat{\boldsymbol{z}}) \otimes \sum_c k(\lambda, F_a(c))}_{\text{Active Perturbation}} + \underbrace{(\gamma \boldsymbol{I}) \otimes k(\lambda, F_p)}_{\text{Passive Perturbation}} + \underbrace{n}_{\text{Noise}}, \quad (7)$$

where γ is the intensity scaling factor considering camera gain and screen attenuation. The blur kernels for active and passive TSR perturbations are $\sum_c k(\lambda, F_a(c))$ and $k(\lambda, F_p)$, respectively. The noise is represented by n, which includes both shot noise and read-out noise. $\hat{\boldsymbol{z}}$ (in pixels) represents an active perturbation image on USC projected from the R/G/B subpixel sets on the TSR (in mm). As shown in Figure 3, the camera's imaging process projects three-dimensional objects onto a two-dimensional plane. Given the screen's proximity to the USC, the projection process can be approximated as a conversion from the camera to the pixel coordinate system. Following the perspective camera model [59], we denote this projection as $\hat{\boldsymbol{z}} = \text{Resize}(\boldsymbol{z}, r_0)$, where $\boldsymbol{z}$ represents R/G/B subpixel sets on the TSR, and r_0 is the USC's focal length.

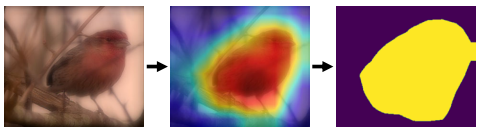


Figure 5: Example of using the Grad-CAM for attacking region estimation: (a) taking an image captured by the USC as input, (b) we leverage the Grad-CAM to generate the heatmap of the most significant region in the image, and then (c) obtain the attacking region.

5 SCREEN-PIXEL PERTURBATION

In this section, we first introduce how to identify the potential attacking region in images captured by USC (Section 5.1). We then present the chromaticity destruction module that selects the optimal color for the screen-pixel perturbation (Section 5.2), followed by the morphology destruction module that determines the optimal position and brightness of the screen-pixel perturbation (Section 5.3). Combining the above two modules, we can activate a single screen-pixel unit on TSR and add corresponding perturbation in images captured by USC, and we call it **one-pixel perturbation**. Lastly, we introduce the **multiple-pixel perturbation** to improve the attacking efficiency by illuminating multiple screen-pixel units simultaneously on TSR (Section 5.4).

Design space. Different from the theoretical model, the design space when generating screen-pixel perturbation on practical under-screen camera smartphone is smaller. Specifically, hardware-related parameters, such as shape, size, and the number of the pixels on the screen are fixed by the manufacturer. Instead, we have the access to each screen-pixel unit in the TSR, and can program the color and brightness of the R/G/B subpixel set in the screen-pixel unit.

5.1 Calculating the Attacking Region

Our goal is to generate adversarial perturbations by modifying as few screen-pixel units as possible (to be less perceptible), while greatly degrading the classification accuracy of the victim deep learning model. To achieve this, we first need to localize the region in the targeted image that has the highest influence on the decision making of the deep learning model. We leverage the Gradient-weighted Class Activation Mapping (Grad-CAM) [50] to estimate the attacking region where the screen perturbation will be added to.

Figure 5 shows the pipeline in calculating the attacking region. Taking an image captured by the USC as input, we

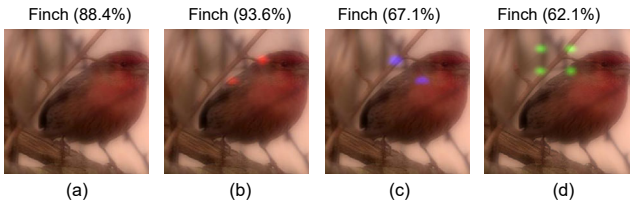


Figure 6: The adversarial strength of the screen-pixel perturbation with different colors: (a) image without active perturbation; by contrast, in (b), (c), and (d), the red, blue, and green subpixel set is used to create the screen-pixel perturbation, respectively.

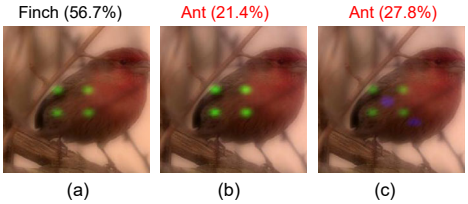


Figure 7: Attacking with different brightness levels and number of subpixel sets used: (a) lighting up the green subpixel set; (c) lighting up the green subpixel set with a high brightness; (d) lighting up both green and blue subpixel sets but with a lower brightness.

first leverage the Grad-CAM to calculate a heatmap of the image based on the surrogate model of the attacker. As shown in Figure 5(b), the heatmap highlights the most significant region in the image that drives the surrogate model to its final prediction. Then, by calculating the average value of the heatmap and thresholding it, we can calculate the attacking region that indicates the potential image area where adversarial perturbations should be added to. Note that for a given image, the heatmap generated by the Grad-CAM varies when different architectures and training datasets are used by the surrogate model. However, because of the transferability effect, different models trained for similar tasks will share highly similar properties and vulnerabilities, even when they have different architectures and are trained on different datasets [11, 40, 41, 55, 71]. Thus, as demonstrated later in Section 6, we can leverage one representative model as the surrogate model for attacking region estimation, but still be able to apply and transfer the resulting adversarial perturbations to different models in the attacking stage.

5.2 Chromaticity Destruction

The ability of an adversarial perturbation to fool the classifier is known as the adversarial strength. In general, for colorization-based perturbation [52], its adversarial strength is proportional to the number of perturbed pixels [36, 42] and the energy change in the pixels [37]. Based on these facts, we introduce the chromaticity destruction to generate the screen-pixel perturbation in three steps.

Step 1: Identifying the dominant color channel. First, we identify the dominant channel of the attacking region. The dominant channel is one of the R/G/B channels with the highest color intensity. Next, for a given screen-pixel unit, we light up one or two of its R/G/B subpixel sets that are distinct from the dominant channel to generate corresponding screen-pixel perturbations. By doing so, the color of the resulting screen-pixel perturbation will be distinct from the dominant channel of the attacking region, which causes the 'color shift' effect on the targeted image with maximized energy change in the perturbed pixels.

As an example, Figure 6 shows the adversarial strength of the screen-pixel perturbations generated by lighting up different subpixel sets. Figure 6(a) exhibits the baseline scenario where no active perturbation has been added. The surrogate classifier utilizes ResNet-50 and is trained on miniImageNet dataset. As shown, the surrogate classifier can correctly recognize the object as 'finch' with a probability score of 88.4%. In Figure 6(b), because R channel is the dominant channel of the attacking region, lighting up the red subpixel set to generate the screen-pixel perturbation does not lead to the expected mis-classification. Instead, the 'finch' can still be correctly recognized with a high probability score of 93.6%. By contrast, as shown in Figures 6(c) and (d), a screen-pixel perturbation generated by lighting up the blue or the green subpixel set can dramatically degrade the probability score of the surrogate model to 67.1% and 62.1%, respectively.

Step 2: Prioritizing R/G/B subpixel sets. As a larger image perturbation has a higher adversarial strength [36, 42, 73], we prioritize the three color subpixel sets in the order of green, blue, and red, according to their actual sizes in the screen-pixel unit. Specifically, the green subpixel set has the highest preference, as the screen-pixel unit in most OLED screens utilizes an RGBG structure, resulting in twice as many green subpixel sets as red and blue subpixel sets (as shown in Figure 2). Thus, green subpixel set can perturb more image pixels and has the largest perturbation size. The blue subpixel set is the second preference, as it has the second-large area. This is because the blue subpixel material has the shortest lifespan, and the area of the blue subpixel set is always maximized to extend the lifespan of the screen [60].

Step 3: Lighting up two subpixel sets. Figure 6 shows that lighting a single subpixel set cannot ensure the perturbation is strong enough to fool the classifier. To improve the adversarial strength, we can either increase the number of perturbed subpixels (by lighting up two subpixel sets simultaneously) or enhance the energy change in the subpixels (by increasing the brightness of the subpixel set). However, as the human visual system is more sensitive to luminance than chrominance, we opt to light up two subpixel sets simultaneously while minimizing their brightness. As an example, Figures 7(a) and (b) show that by increasing the brightness of the green subpixel set, the 'Finch' is misclassified as 'Ant', but the perturbation becomes more perceptible. By contrast, Figure 7(c) shows that lighting up both green and blue subpixels at a lower brightness can still fool the classifier.

5.3 Morphology Destruction

Below, we introduce the morphology destruction, which optimizes the location and brightness of the screen-pixel perturbation. We use the following notations in our design.

- $\boldsymbol{I}$: is the targeted pristine image with true label l.
- $\boldsymbol{z}$: is the original configuration of the unperturbed screen-pixel units in the translucent screen region.
- $\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}$: the configuration of the screen-pixel units when generating the adversarial perturbation, which is configured by screen-pixel index j, color vector $\boldsymbol{c}$, and subpixel brightness vector $\boldsymbol{b}$. Specifically, j indicates that the jth screen-pixel unit in the TSR will be manipulated; $\boldsymbol{c}$ is a three-dimensional vector that indicates which of the three R/G/B subpixel sets will be light up; similarly, $\boldsymbol{b}$ is a three-dimensional vector that indicates the brightness of the three subpixel sets.
- $\oplus$: denotes the image formation process of the under-screen camera, which is defined in Equation (7).
- Φ: denotes the surrogate classifier.
- $\Phi(\boldsymbol{I}, l)$: denotes the probability of surrogate classifier Φ in classifying the image $\boldsymbol{I}$ with label l.

The goal of the morphology destruction is to search for a configuration $\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}$ that can fool the surrogate classifier Φ. As we only change a single screen-pixel, we aim to find the screen-pixel index j such that the surrogate classifier is most likely to misclassify the object of interest. Assuming a perturbation at the jth screen-pixel, we need the probability of an incorrect label surpassing the true label l to trick Φ. To reach the goal, we aim to find the most susceptible label when the perturbation is added at jth screen-pixel. In particular, we use the growth rate of the output probability of the surrogate classifier Φ to measure the susceptibility for each incorrect label $\hat{l}$ ($\hat{l} \neq l$). Our intuition is that the output probability of the surrogate classifier Φ for an incorrect label is more likely to increase more when it has a larger growth rate. Formally, the growth rate for $\hat{l}$ can be computed as follows:

$$s(\hat{l}, j) = \frac{\Phi(\boldsymbol{I} \oplus \boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}, \hat{l}) - \Phi(\boldsymbol{I} \oplus \boldsymbol{z}, \hat{l})}{\Phi(\boldsymbol{I} \oplus \boldsymbol{z}, \hat{l})}, \tag{8}$$

where $s(\hat{l}, j)$ is the growth rate; term $\boldsymbol{I} \oplus \boldsymbol{z}$ is the image captured by the under-screen camera with unperturbed screen configuration $\boldsymbol{z}$; term $\boldsymbol{I} \oplus \boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}$ is the image captured with the perturbed configuration $\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}$; $\boldsymbol{c}$ is calculated by the chromaticity destruction, which indicates the lighting of the R/G/B subpixel sets given input $\boldsymbol{I} \oplus \boldsymbol{z}$; $\boldsymbol{b}$ is set to a low brightness level of $\boldsymbol{b}_0 = 0.01$. Given the growth rate for each incorrect label, we view the label whose growth rate is the largest as the most susceptible label, i.e., $\tilde{l}_j = \text{argmax}_{\hat{l} \neq l}\, s(\hat{l}, j)$.

Recall that our goal is to find the screen-pixel index j that is most likely to cause the misclassification of the surrogate classifier. We reach the goal by finding the j whose most susceptible label has the largest growth rate. Formally, we can find j by solving the following optimization problem:

$$j = \text{argmax}_{\hat{j}}\, s(\tilde{l}_{\hat{j}}, \hat{j}), \tag{9}$$

where $s(\tilde{l}_{\hat{j}}, \hat{j})$ is the growth rate of the most susceptible label $\tilde{l}_{\hat{j}}$ when the perturbation is added at $\hat{j}$th screen-pixel.

After finding the optimal index j, we calculate the minimum required brightness $\boldsymbol{b}$ by a one-directional search process. Initially, we set the brightness of the screen-pixel unit to a small value $\boldsymbol{b} = \boldsymbol{b}_0$. We then gradually increase $\boldsymbol{b}$ with a small step size $\Delta\tau$. The search is terminated when:

$$\Phi(\boldsymbol{I} \oplus \boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}, l) < \Phi(\boldsymbol{I} \oplus \boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}, \hat{l}), \exists \hat{l} \neq l, \text{ or } \boldsymbol{b} > \varepsilon, \tag{10}$$

which means either the perturbation generated by $\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}$ is able to fool the surrogate classifier with brightness $\boldsymbol{b}$ or the maximum screen-pixel intensity budget ε is reached. This process repeats for each subpixel set of the chosen colors until the optimal set requiring the least brightness is found.

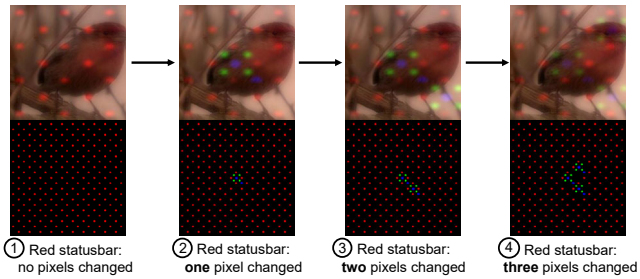


Figure 8: Example of pixel spread process: transitioning from no manipulated screen-pixel units to activate three screen-pixel units.

5.4 Multiple-Pixel Perturbation

We are motivated by the fact that when using the smartphone and looking at the screen at a certain distance, the human visual system has a high tolerance in screen pixel changes. Current research has shown that a change of 120 pixels is indistinguishable [27] per degree of viewing angle when viewing at a normal distance. Thus, instead of lighting up single screen-pixel unit, we can potentially manipulate multiple screen-pixel units to enable a larger perturbation area. This can greatly improve the adversarial strength of the generated screen-pixel perturbation while ensuring the changes on the translucent screen area are unnoticeable.

First, we introduce the process of selecting multiple screen-pixel units to generate the perturbation. As shown in Figure 8, we apply the morphology destruction method (in Section 5.3) to determine the position and brightness of multiple screen-pixel units for perturbation generation when the one-pixel perturbation fails. The added screen-pixel units are adjacent to the screen-pixel unit selected for the one-pixel perturbation due to the regional aggregation effect [64].

In one-pixel perturbation, the color of subpixel sets is selected based on the dominant color channel of the captured image. However, this strategy may not always be applicable, as the object of interest can be obscured by the color

of the screen. For instance, when the screen displays a red status bar, the object of interest will be concealed by red screen-pixel, and makes it difficult to discern the original dominant color channel of the object. To solve this problem, we propose the *differential incremental module*, which estimates the impact of each individual R/G/B subpixel set on the adversarial strength of the generated perturbation.

Specifically, we add a small increment to the brightness vector by $\boldsymbol{b}^{'} = \boldsymbol{b} + \Delta\tau$ for each R/G/B subpixel set. Then, we select the colors of subpixel sets perturbation (c_1/c_2) by:

$$c_1/c_2 = \begin{cases} G/B,\ R = \arg\min_{c\in\{R,G,B\}} \left(\frac{\Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}^{'}},\hat{l})}{\Phi(\boldsymbol{I}\oplus\boldsymbol{z},\hat{t})}\right), \\ G/R,\ B = \arg\min_{c\in\{R,G,B\}} \left(\frac{\Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}^{'}},\hat{l})}{\Phi(\boldsymbol{I}\oplus\boldsymbol{z},\hat{t})}\right), \\ B/R,\ G = \arg\min_{c\in\{R,G,B\}} \left(\frac{\Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}^{'}},\hat{l})}{\Phi(\boldsymbol{I}\oplus\boldsymbol{z},\hat{t})}\right). \end{cases} \quad (11)$$

By adding the increment $\Delta\tau$ to the corresponding screen pixel color $\boldsymbol{c}$, we can identify the two subpixel sets that have the highest impact on the adversarial strength. Additionally, we use a second-order difference for brightness calculation, instead of the linear search used for the one-pixel perturbation. Specifically, we use the second-order difference of the probability score as the searching condition:

$$\Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}^{''}},\hat{l}) - \Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}^{'}},\hat{l}) < \Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}^{'}},\hat{l}) - \Phi(\boldsymbol{I}\oplus\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}},\hat{l}), \quad (12)$$

where $\boldsymbol{b}^{'} = \boldsymbol{b} + \Delta\tau$ and $\boldsymbol{b}^{''} = \boldsymbol{b} + 2\Delta\tau$. If the second-order difference satisfies the above condition, we stop increasing the current screen-pixel unit's brightness and start calculating the next selected screen-pixel unit's position and brightness.

Finally, to ensure the changes in the multiple screen-pixel units are less perceptible by human visual system, we add a constraint on Equation (12) when searching $\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}$:

$$\text{DSSIM}(\text{Resize}(L(\boldsymbol{z}),\alpha), \text{Resize}(L(\boldsymbol{z}_{j,\boldsymbol{c},\boldsymbol{b}}),\alpha)) < \rho, \quad (13)$$

where DSSIM$(\cdot)$ calculates the structural dis-similarity index, a common measure of user-perceived image distortion [53, 62]. Resize$(\cdot)$ and $L(\cdot)$ represent image resizing and low-pass filtering functions, used to simulate the pixel density and viewing distance changes during smartphone use, and to determine the visual impact of screen pixel changes on human perception. Lastly, ρ denotes the perceptual perturbation budget. We leverage the downsampling scale α to simulate human visual perception, influenced by the viewing angle, screen resolution, and viewing distance [8, 22].

6 EVALUATION

6.1 Methodology

Goal and metrics. For simplicity, we evaluate our method from the attacker's perspective: the aim is to manipulate TSR's screen-pixel units, generating perturbed images to fool different machine learning models, i.e., **target models**. The screen-pixel perturbation's performance is measured by the target model's accuracy in classifying perturbed images.

Applications and target Models. We consider two widely used daily image-based applications.

- **Image classification**. We consider three DNN architectures with six models: (1) **ResNet** [24], which includes ResNet-18 and ResNet-50; (2) **MobileNet** [25], which includes MobileNet V3 Large (MobileNet$_\text{L}$) and Small (MobileNet$_\text{S}$); and (3) **ShuffleNet V2** [34], which has two variants ShuffleNet V2 with half of the network parameters (ShuffleNet$_\text{S}$) and with all network parameters (ShuffleNet$_\text{L}$). All models are pre-trained on the ImageNet dataset [46].
- **Face recognition**. We consider two representative backbone models **IncepResNet V1** [57] (IncepResNet) and **MobileNet V2** [48] (MobileNet) that are pre-trained on the VGGFace2 dataset [6] and WebFace dataset [70], respectively.

Synthesized Datasets. We leverage the theoretical model to embed screen perturbations to images from two datasets. Specifically, we use the **miniImageNet** dataset [61], with 60,000 images across 100 classes, to evaluate the image classification task. We use the **FaceScrub** dataset [38], with 38,202 images from 530 people, to evaluate the face recognition task. We randomly select 500 and 530 images from miniImageNet and FaceScrub, respectively, to add screen perturbations. In simulation, our model employs a TSR on OLED screens with a pixel density of 400 Pixels Per Inch (PPI), utilizing a circular R/G/B subpixel shape that completely fills the screen-pixel area. To ensure consistency with the USC in Commercial Off-The-Shelf (COTS) smartphones, we utilize camera parameters that are identical to those employed in commercial devices. We also incorporate peak wavelength values for the R, G, and B channels based on prior research [67, 75] with values of 0.61 μm, 0.53 μm, and 0.47 μm, respectively.

Dataset collected by testbed. We setup a testbed to acquire practical USC images. The setup is shown in Figure 9, which consists of a 4K LCD monitor displaying pristine images, and three COTS USC smartphones, i.e., Samsung Fold4, ZTE AXON30, and Xiaomi MIX4. The pixel density of the TSR in these devices is 400 PPI. The size of the TSR is 58∗58 pixels, 108∗60 pixels, and 108∗60 pixels for the Fold4, AXON30, and MIX4, respectively. A custom Android application is developed to control the screen-pixel units for perturbation generation. The smartphone is positioned at the center of the monitor and is adjusted to cover the monitor's full range.

We then leverage high-resolution full-face images from the XGaze dataset [72] to generate perturbed images. Specifically, we select a subset of 12,720 images from 110 subjects, and randomly sample a total 549 images to add screen perturbations. The selected images are displayed on the 4K monitor in full-screen mode and adjusted to maintain the aspect ratio through rotation or resizing. As an example, Figures 9(b-d) shows the perturbed images captured by the three smartphones with one-pixel perturbation added. The resulting

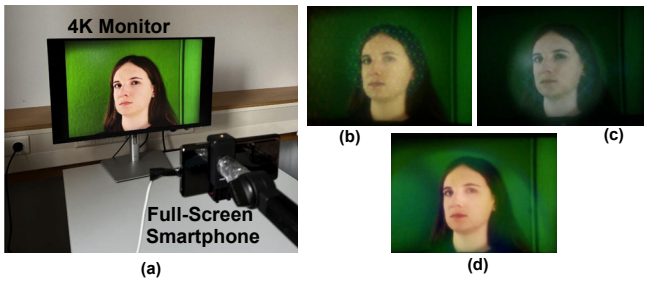


Figure 9: The testbed setup: we use COTS smartphones to generate images containing screen perturbations, and images captured by (b) ZTX AXON30, (c) Xiaomi MIX4, and (d) Samsung Fold4, respectively.

Figure 10: Screenshots of the smartphone status bar when running the one-pixel perturbation on (a) ZTE AXON30, (b) Xiaomi MIX4, and (c) Samsung Fold4. The TSR is highlighted by the red dotted rectangle, and magnified in (d). These changes are imperceptible.

screen-pixel perturbations differ as different screen layouts. Moreover, Figure 10 shows the screenshots of the status bar when the three smartphones are generating one-pixel perturbation. As highlighted in the red dotted rectangle, the changes of screen-pixel units in the TSR are imperceptible.

6.2 Performance on Synthesized Datasets

Performance on image classification. We first investigate the effectiveness of screen perturbation in disrupting image classification on the miniImageNet dataset. We consider five different scenarios, including (1) clean image with no screen perturbation, (2) passive perturbation when the status bar is black, (3) passive perturbation when the status bar is white, (4) passive perturbation with the additional active perturbation generated by one-pixel perturbation, or by (5) multiple-pixel perturbation. The results are shown in Table 1. Even with a black status bar, the accuracies of all models are decreased by 20% due to the passive perturbation. Furthermore, with a white status bar, the accuracy of all models decreased by 30%, due to the uncontrolled active perturbation. In comparison, when applying the one-pixel and multiple-pixel perturbation, the accuracies of all examined models are degraded below 20% and 10%, respectively. The results demonstrate the effectiveness of the proposed screen-pixel perturbation in disrupting image classification.

Performance against deblurring algorithms. We also evaluate the performance when state-of-the-art deblurring method is applied. Specifically, we use the unsupervised Wiener filter to alleviate the blurring effect caused by screen passive perturbation [67, 75]. The results are shown in Table 1. The deblurring method can effectively eliminate the impact from the passive perturbations, i.e., the accuracy of all examined models is increased after deblurring when there is only passive perturbation on the image. However, the deblurring method does not help when one-pixel or multiple-pixel perturbations have been applied, i.e., only a modest 5% accuracy improvement after applying the deblurring method.

Transferability. The perturbation should be effective even when the target model is different from the surrogate model. Current work [11] suggests that transferability of perturbation between models depends on the robustness of the surrogate model used to create it, and more robust surrogate models are less reactive to small perturbations. Thus, to ensure the transferability of the screen perturbations, we first retrain the surrogate model using perturbed images generated from a white status bar. This provides the surrogate model with exposure to fixed screen perturbation. We then use the updated surrogate models to generate multiple-pixel perturbations on the miniImageNet dataset. The results shown in Table 2 demonstrate that the multiple-pixel perturbations transfer almost perfectly across different models.

Ablation Study. First, we investigate the impact of the maximum screen-pixel intensity budget (ε) on the performance of the one-pixel perturbation attack (in Equation (10)). The initial intensity of the screen-pixel is 0.01 and the step size of each iteration is 0.01. Then, we vary the maximum screen-pixel intensity constraint from 0 to 1. The results are shown in Figure 11(a). the accuracy gradually decreases with the increase of the maximum screen-pixel intensity constraint. Even when the maximum screen-pixel intensity is limited to 0.2, the accuracy still drops by more than 30%.

Next, we examine the step size ($\Delta\tau$) effect on the screen-pixel intensity search (in Equation (10)), determining the optimal intensity for one-pixel perturbation. We fix the maximum screen-pixel intensity to 1 and the initial screen-pixel intensity to 0.01, and then vary the step size. Results in Figure 11(b) show image classification accuracy gradually increases with the step size, as a larger step size complicates finding the optimal intensity. To reduce the computational overhead, we employ a step size of 0.01 in following experiments.

We also investigate the impact of initial screen-pixel intensity (b_0) (in Equation (8)) on the results of one-pixel perturbation, by fixing the maximum pixel intensity and step size to 1 and 0.01, respectively, and varying the initial intensity of the screen-pixel. As shown in Figure 11(c), the variation in initial intensity has a negligible impact on the final performance. We chose to use a smaller initial intensity of 0.01.

Finally, we evaluate the impact of the DSSIM in multiple-pixel perturbation (in Equation (13)). The results are shown in Figure 11(d). As the DSSIM perturbation budget (ρ) increases, the accuracy decreases. Specifically, when ρ is set to 0.05, the accuracy drops to the same level as that of one-pixel perturbation, at around 20%. Furthermore, when ρ exceeds

Table 1: Performance of both one-pixel and multiple-pixel perturbations on miniImageNet.

Target model	Clean image	Black screen Blurred/Deblurred	White screen Blurred/Deblurred	One-pixel Blurred/Deblurred	Multiple-pixel Blurred/Deblurred
ResNet-18	87.4%	62.4% / 86.8%	56.8% / 75.2%	**16.6%** / **20.6%**	**5.0%** / **7.6%**
ResNet-50	94.2%	74.4% / 94.6%	66.8% / 86.2%	**20.8%** / **26.8%**	**5.8%** / **9.2%**
$\text{MobileNet}_\text{S}$	82.4%	58.8% / 81.6%	50.6% / 70.2%	**10.2%** / **14.8%**	**5.4%** / **10.0%**
$\text{MobileNet}_\text{L}$	91.6%	69.8% / 90.8%	66.2% / 79.2%	**13.4%** / **17.4%**	**6.4%** / **11.4%**
$\text{ShuffleNet}_\text{S}$	71.0%	49.8% / 70.2%	29.6% / 46.0%	**7.2%** / **9.0%**	**4.6%** / **5.4%**
$\text{ShuffleNet}_\text{L}$	85.4%	64.4% / 85.2%	47.8% / 62.2%	**16.0%** / **21.0%**	**5.8%** / **6.2%**

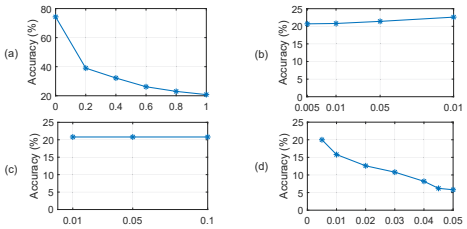


Figure 11: Accuracy vs. (a) maximum intensity, (b) step sizes, (c) initial intensity, and (d) DSSIM budget.

Table 2: Performance (attack transferability) of multiple-pixel perturbations on miniImageNet.

Robust Surrogate Model	Target Model: ResNet-18 Blurred/*Deblurred*	ResNet-50 Blurred/Deblurred	$\text{MobileNet}_\text{S}$ Blurred/Deblurred	$\text{MobileNet}_\text{L}$ Blurred/Deblurred	$\text{ShuffleNet}_\text{S}$ Blurred/Deblurred	$\text{ShuffleNet}_\text{L}$ Blurred/Deblurred
ResNet-18	**4.8%** / **7.4%**	6.8% / 12.2%	5.6% / 7.8%	7.8% / 12.2%	4.2% / 4.2%	7.4% / 8.8%
ResNet-50	5.0% / 7.6%	**5.8%** / **9.2%**	5.4% / 10.0%	6.4% / 11.4%	4.6% / 5.4%	5.8% / 6.2%
$\text{MobileNet}_\text{S}$	6.4% / 11.6%	7.8% / 16.4%	**4.0%** / **9.2%**	8.4% / 16.2%	4.6% / 7.4%	6.8% / 11.0%
$\text{MobileNet}_\text{L}$	6.4% / 7.6%	5.0% / 10.6%	3.4% / 6.8%	**6.8%** / **10.2%**	4.4% / 5.0%	6.4% / 7.8%
$\text{ShuffleNet}_\text{S}$	8.2% / 12.8%	11.2% / 23.2%	7.4% / 14.6%	11.6% / 22.6%	**7.4%** / **8.8%**	9.0% / 15.6%
$\text{ShuffleNet}_\text{L}$	7.4% / 9.8%	9.4% / 16.0%	6.4% / 11.2%	8.2% / 14.4%	6.2% / 7.4%	**6.8%** / **9.8%**

0.03, the accuracy of multiple-pixel perturbation falls below 10%. Existing work [30] suggests that higher DSSIM values (up to 0.2) are imperceptible to human eyes.

Performance on face recognition using FaceScrub. We also evaluate our method in face classification task using the FaceScrub dataset. The results are shown in Table 3. The passive screen perturbation leads to a substantial accuracy degradation of up to 40% and 60% for IncepResNet and MobileNet, respectively. Both one-pixel and multiple-pixel perturbation can decrease the accuracy of all models to less than 1%. Moreover, we also apply the deblurring method to mitigate the adverse impact of the screen perturbations. The deblurring method can mitigate the impact of passive screen perturbations, but fails to remove the impact of the proposed one-pixel and multiple-pixel perturbations.

6.3 Performance on Testbed Dataset

Overall performance. Table 4 shows that proposed one-pixel perturbation can reduce the accuracy of both IncepResNet and MobileNet to less than 5%. The multiple-pixel perturbation further reduces the accuracy of both IncepResNet and MobileNet to under 1%, demonstrating the effectiveness of our proposed screen perturbation methods in real-world scenarios. The results in Section 6.2 and Section 6.3 align with each other, as shown in Table 3 and Table 4.

Different color in status bar. We also evaluate the multiple-pixel perturbation with four different status bar colors: *red, blue, green, white*. Note that when the status bar is displayed in red, green, or blue, only the corresponding primary color subpixel sets are illuminated. When the status bar is white, all three primary color subpixel sets are illuminated. The results are presented in Figure 12(a). Without active perturbation, we can achieve 40% to 50% of accuracy in all four background colors. With the multiple-pixel perturbation, the face recognition accuracy drops to 1% for all four colors.

Screen Diversity. We evaluate one-pixel perturbation with different smartphones. The results are shown in Figure 12(b). In the absence of screen-pixel perturbation and relying solely on the passive perturbation, the performance of images captured by under-screen cameras from different smartphone manufacturers is relatively similar, hovering around 80%. However, after adding one-pixel perturbation, the classification performance of under-screen camera captured images dropped below 20%. Specifically, the classification accuracy of ZTE AXON30 declined the most, falling below 5%, while that of Samsung Fold4 decreased the least. This is because the pixel size of the Samsung translucent screen region is too large, causing the screen-pixel perturbation on the under-screen camera captured image to be too diffuse, thereby reducing the perturbation intensity of the attacking region.

Image Quality. We further evaluate the impact of screen-pixel perturbation on image quality by comparing a set of unperturbed images with the corresponding perturbed images, all captured using the ZTE AXON30. We use two widely adopted metrics: Peak Signal-to-Noise Ratio (PSNR) [59] and Structural Similarity Index Measure (SSIM) [63]. Specifically, PSNR quantifies the variations in individual pixel intensity levels, while SSIM assesses the structural distortions within an image, e.g., stretching, banding, and twisting. The results are shown in Table 5. For images containing one-pixel perturbations, the PSNR and SSIM are 23.72 dB and 0.84, respectively. For images added with multiple-pixel perturbations, the PSNR and SSIM are 21.5 dB and 0.8, respectively, as more screen-pixel perturbations have been added to the captured images. Note that for images with acceptable viewing quality, the minimum PSNR and SSIM are in the range of 20∼40 and

Table 3: Performance of both one-pixel and multiple-pixel perturbations on Facescrub dataset.

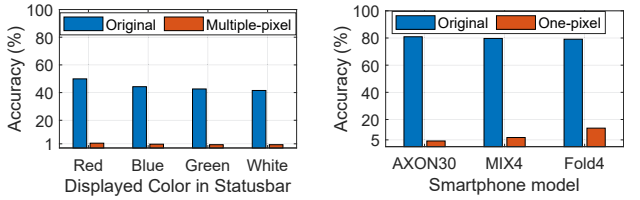
Model	Clean image	Black screen Blurred/Deblurred	One-pixel IncepResNet Blurred/Deblurred	One-pixel MobileNet Blurred/Deblurred	White screen Blurred/Deblurred	Multiple-pixel IncepResNet Blurred/Deblurred	Multiple-pixel MobileNet Blurred/Deblurred
IncepResNet	90.57%	50.75% / 85.09%	**0.00%** / **0.76%**	0.38% / 0.38%	38.30% / 72.64%	**0.00%** / **0.00%**	0.00% / 0.00%
MobileNet	84.53%	26.04% / 81.89%	0.19% / 0.76%	**0.57%** / **0.76%**	19.25% / 58.11%	0.00% / 0.00%	**0.00%** / **0.00%**

Table 4: Performance of both one-pixel and multiple-pixel perturbations on xGaze dataset.

Model	Clean image	Black screen	One-pixel IncepResNet	One-pixel MobileNet	White screen	Multiple-pixel IncepResNet	Multiple-pixel MobileNet
IncepResNet	94.72%	81.06%	**4.01%**	2.37%	41.53%	**0.36%**	0.91%
MobileNet	94.54%	70.13%	2.37%	**1.28%**	34.43%	0.18%	**0.55%**

Table 5: The PSNR and SSIM when one-pixel and multiple-pixel perturbations are embedded in the image.

Metrics	One-pixel	Multiple-pixel
PSNR	23.72 dB	21.50 dB
SSIM	0.84	0.80



(a) Accuracy vs. status bar color (b) Accuracy vs. smartphone model

Figure 12: Accuracy under different hardware setups.

0.8~0.9, respectively [33, 44]. Thus, the perturbed images have acceptable image quality.

6.4 User Study

Below, we conduct a user study to investigate if the added perturbation is visible in images captured by the under-screen cameras. We also examine if and how changes in the smartphone TSR affect the user's experience with the smartphone use. We recruit 30 participants from our university (13 female and 17 male, aged between 20 and 45) to conduct a user study. We advertised our study through university mailing lists, social networks, and advertising boards in our department building. All participants had little or no experience with USC smartphones. Ethical approval for carrying out the user study has been granted by our organization.

Study design. We design two tasks in the study.

• **Perturbed images task**. We investigate if users perceive the screen perturbation added to images. Using the three USC smartphones from Section 6.3, Samsung Fold4, ZTE AXON30, and Xiaomi MIX4, we prepare two image sets: one-pixel and multiple-pixel paired sets. We randomly select ten different subjects from the XGaze [72] dataset each time. In the one-pixel paired set, each smartphone captures ten *unperturbed images* without adding any screen perturbation and ten *perturbed images* with one-pixel perturbation from these subjects. The multiple-pixel paired set consists of ten *unperturbed images* captured by AXON30 when the status bar is in red/green/blue/white, and ten *perturbed images* captured by AXON30 with added multiple-pixel perturbation. Images in two image sets are displayed on a 27-inch monitor in a random order, and each image is played for 10 seconds.

Results. Participants answer two 5-point Likert scale questions (1: strongly disagree; 5: strongly agree) after viewing each displayed image:

- Q1: *Can you see a person in this image?*
- Q2: *Do you think there is a perturbation in this image?*

A standard two-sample t-test [47] is conducted to compare user visual perception for *unperturbed* and *perturbed* image sets. The null hypothesis (H_0) posits no true difference between the means of the two sets, while the alternate hypothesis (H_a) suggests a non-zero difference. Setting a significance level $\alpha = 0.05$, we calculate p to get the t-test result, where p is the likelihood that the observed difference is occurred by chance. If $p > \alpha$, we accept H_0 and conclude that participants have similar responses on both sets. Otherwise, we reject H_0 at the significance level of 0.05, and conclude that participants have different responses on the two sets. The t-test results are shown in Tables 6 and 7. H_0 is all accepted in Q1, showing similar user perception for seeing a person in all unperturbed and perturbed images. In Q2, images captured by Fold 4 accept H_0, suggesting less perceptible one-pixel perturbation, while images from AXON30 and MIX4 reject it, indicating more visible perturbations due to their different TSR designs. As shown in Figure 9, different TSR designs of smartphones result in diverse one-pixel perturbation patterns. However, images captured with different status-bar colors accept H_0, suggesting participants cannot perceive the added multiple-pixel perturbation since the color of the status bar has added a fixed perturbation to captured images.

• **Smartphone usage task**. Next, we investigate the perceptibility of screen-pixel changes in the TSR and their impact on user experience during typical smartphone usage. We adopt the three USC smartphones used in Section 6.3, and randomly assign ten participants to use each of the smartphones. We developed two smartphone apps, i.e., *image app* and *text app*, for image and text viewing. To reduce individual differences in visual perception of screen perturbations, we conduct a within-subject study [9] and consider two screen settings: (1) *with screen perturbation*, and (2) *without screen perturbation*. We ask the participant to use the two developed

Table 6: T-test results of the one-pixel paired set.

Device	AXON30		MIX4		Fold4	
Question	Q1	Q2	Q1	Q2	Q1	Q2
p	0.8822	2.83×10^{-7}	0.3416	2.67×10^{-5}	0.3367	0.0545
Results	H_0	H_a	H_0	H_a	H_0	H_0

Table 7: T-test results of the multiple-pixel paired set.

Color	Red		Green		Blue		White	
Question	Q1	Q2	Q1	Q2	Q1	Q2	Q1	Q2
p	0.1326	0.4063	0.7063	0.8051	0.6367	0.9484	0.3007	0.2298
Results	H_0	H_0	H_0	H_0	H_0	H_0	H_0	H_0

apps on the assigned smartphone, and each app is used for two minutes. We configure the app with both screen settings and load them in random order, thus ensuring one-minute usage for each screen setting. We consider both one-pixel and multiple-pixel perturbations in the experiment and modify the screen pixel in the TSR accordingly.

Results. We provide a short questionnaire to the participants, and ask them if they have noticed any screen perturbation during the use of the apps. The results show that none of the 30 participants notice the added one-pixel or multiple-pixel screen perturbation. We also conduct a standard two-sample t-test [47] to investigate the visual perception of the user in the two screen settings, i.e., with and without screen perturbation added. The result shows that H_0 is accepted with $p > 0.99$, which indicates that two screen settings on *image app* and *text app* have the same visual perception.

6.5 Countermeasures

Below, we investigate possible countermeasures against the proposed method. Many defenses [10, 20, 23, 29, 35, 43] have been proposed to defend against adversarial perturbations. Depending on whether those defenses have formal robustness guarantees, we can categorize them into *empirical defenses* [20, 23, 35] and *certified defenses* [10, 29, 45]. We consider state-of-the-art defenses from both categories. For empirical defenses, we consider adversarial training [35] as it is viewed as one of the most effective empirical defenses. For certified defenses, we consider randomized smoothing [10] as it is applicable to any classifier and scalable to large DNN.

Adversarial training. Adversarial training [20, 35] leverages adversarially perturbed training examples to train a model to enhance its robustness. First, we generate adversarial examples using the Project Gradient Descent (PGD) attack [35]. We employ an L_∞-based PGD attack on the XGaze dataset, with a maximum distortion of 0.1 and an untargeted attack mode. We run the PGD attack for 50 steps with a step size of 0.01. We train IncepResNet-V1 and MobileNet-V2 for 200 epochs with a learning rate of 0.1, decayed by 0.1 after 100 and 150 epochs, respectively [35, 39]. Subsequently, we evaluate the performance of the resulting models against one-pixel and multiple-pixel perturbations using the XGaze dataset [72]. The results are shown in Table 8, with adversarial training, the classification accuracy of the two DNN models has still degraded below 8% and 4% when one-pixel and multiple-pixel perturbations have been added.

Table 8: Accuracy of one-pixel and multiple-pixel perturbations with adversarial training on XGaze dataset.

Model	Clean image	One-pixel		Multiple-pixel	
		IncepResNet	MobileNet	IncepResNet	MobileNet
IncepResNet	85.79%	**7.29%**	4.19%	**3.83%**	1.82%
MobileNet	81.42%	8.20%	**4.92%**	4.00%	**1.64%**

Table 9: Accuracy of one-pixel and multiple-pixel perturbations when adversarial training is enhanced with perturbed images.

Model	Clean image	One-pixel		Multiple-pixel	
		IncepResNet	MobileNet	IncepResNet	MobileNet
IncepResNet	87.07%	**10.56%**	5.83%	**4.37%**	2.00%
MobileNet	86.52%	10.20%	**5.28%**	1.82%	**2.37%**

Moreover, we also perform adversarial training with a dataset that comprises clean images paired with corresponding perturbed images containing one-pixel perturbation. The results are presented in Table 9. When comparing with the results reported in Table 4, we can observe a decline in classification accuracy when perturbed images are utilized for training. However, the accuracy is below 10% and 5%, respectively, when the captured images contain one-pixel or multiple-pixel perturbations. Overall, the results indicate that adversarial training is not sufficient to mitigate the perturbations added by the proposed method. While this doesn't conclusively negate the possibility of developing robust models, it indicates the inherent challenges in such pursuits.

Randomized smoothing. Given an arbitrary classifier H (called *base classifier*) and a testing input $\mathbf{x}$, randomized smoothing builds a certifiably robust smoothed classifier G by adding zero-mean isotropic Gaussian noise $\mathcal{N}(0, \sigma^2\mathbf{I})$ to the testing input $\mathbf{x}$, where σ is the standard deviation and $\mathbf{I}$ is the identity matrix. Formally, the predicted label of the smoothed classifier G for the testing input $\mathbf{x}$ is $G(\mathbf{x}) = \text{argmax}_{c=1,2,\cdots,C} \Pr(H(\mathbf{x} + \mathcal{N}(0, \sigma^2 I)) = c)$, where C is the total number of classes. Existing work [10] shows that the predicted label of the smoothed classifier G for the testing input $\mathbf{x}$ does not change when the adversarial perturbation added to $\mathbf{x}$ is bounded. To compute $G(\mathbf{x})$ in practice, randomized smoothing first adds random Gaussian noise to the testing input $\mathbf{x}$ to create M noisy versions of the testing input, then use the base classifier H to predict labels for those M noisy inputs, and finally take a majority vote over the M predicted labels as the final prediction for the testing input. Following previous work [10], we set $\sigma = 0.5$ and $M = 10^5$. Moreover, we train the base classifier on training inputs augmented with Gaussian noise to improve the robustness of the smoothed classifier. Table 10 shows our experimental results, which show that the classification accuracy of the smoothed classifier built by randomized smoothing is still very low. The reason is that randomized smoothing can only certify a very small perturbation. Our results demonstrate that randomized smoothing is insufficient to mitigate our proposed one-pixel and multiple-pixel perturbations.

Table 10: Accuracy of one-pixel and multiple-pixel perturbations under randomized smoothing.

Model	Clean image	One-pixel		Multiple-pixel	
		IncepResNet	MobileNet	IncepResNet	MobileNet
IncepResNet	93.99%	**6.01%**	3.46%	**0.73%**	0.36%
MobileNet	94.90%	6.19%	**3.64%**	1.09%	**0.91%**

7 LIMITATIONS AND DISCUSSIONS

Difference between attacking camera input & output. Directly modifying captured images is challenging for attackers, as it requires write access from smartphone OS to a set of media files [15]. However, attacking the optical path (camera input) is relatively easier, as Android does not have permission restrictions on UI settings, allowing any app to set the screen content and potentially perturb camera's input.

Practical deployment. In typical smartphone usage scenarios, successive frames captured by the camera are similar. Thus, we only need to compute the proposed screen-pixel perturbation once, rather than for each image separately, making the attack more feasible.

Requirement of uncorrupted images. The current design of our system requires an uncorrupted image to compute the attacking region for the screen perturbation. This image can be obtained from many sources, e.g., selfies of the subject published on social media, and does not need to be taken in run-time. For instance, with the aim to defend against unauthorized face recognition systems, users can take a selfie in run-time or upload a previously taken selfie to compute the attacking region. Since the face of the subject is always located in the center of the selfie, it leads to similar attacking regions as long as the image contains the same subject. Similarly, in the attacking scenario, malicious parties can find and leverage the selfie of the victim published on social media to achieve the same purpose. For future research direction, we can release the requirement of an uncorrupted image by designing a universal and scene-independent perturbation.

Potential operating system-level defense. A potential defensive strategy is to prohibit a single application from rendering content on the screen and accessing the camera simultaneously. However, it is inconvenient for users, especially when they need to use video communication applications that usually require camera access, together with other applications. In fact, many modern smartphones, e.g., Samsung Fold3/4, ZTE AXON 20/30/40, Xiaomi MIX4, and K50S, are designed to support simultaneous operation of one app's camera usage while another app utilizes the screen.

8 RELATED WORK

Image processing for the under-screen camera. There are efforts in modeling and restoring images that are affected by passive perturbations caused by the translucent screen [17, 51, 56, 67, 75]. Specifically, Zhou et al. [75] and Yang et al. [67] modeled the passive perturbation of the screen and proposed an unsupervised Wiener deconvolution method. Moreover, they employed deep neural networks to address the issues of large blur and low signal-to-noise ratio in under-screen images [51, 56, 75]. Feng et al. [18] and Gao et al. [19] explored simulation pipelines to generate synthetic datasets with real-captured passive perturbation. While existing studies focused solely on passive perturbations, we are the first to consider the active perturbations caused by the screen displays. We apply the Huygens-Fresnel principle to model the impact of different color pixels and screen parameters on image formation. This allows us to analyze the impact of active screen perturbations on deep neural networks.

Programmable aperture. Our work is related to the concept of programmable aperture that has been implemented in some new types of cameras such as DiffuserCam [4] and FlatCam [5]. Existing works of programmable aperture leverage amplitude or phase masks to code the aperture of a camera lens, and operate at scales that are larger than screen pixels. This is different from the under-screen camera apertures we consider, which coexist with R/G/B OLED arrays and can be dynamically manipulated to generate screen-pixel perturbation on the TSR when the screen is illuminated.

Adversarial attacks can be categorized into digital space attacks [7, 35, 58] or physical space attacks [16, 26, 32, 54]. Digital space attacks directly change pixel values in the digital pixel domain, typically using methods such as PGD [35] and C&W [7]. However, they may not generalize to real-world scenarios due to the constraints present in the physical environment. Physical attacks, such as those utilizing graffiti [16], rectangular stickers [26], eyeglasses [54], or LED lamps [49, 76], have been explored for privacy protection but are often limited by artificial settings or hardware modifications. By contrast, ours is the first to generate adversarial perturbation on USC-captured images utilizing both passive and active screen perturbation of full-screen devices. Our threat model is unique in that we inject a perturbation into the optical path between the camera and the object, disrupting any object without tampering with the object itself.

9 CONCLUSION

We show for the first time the screen can be used for adversarial attack and defense on under-screen camera. We identify the activated screen-pixel that can be exploited for applying perturbation on images captured by under-screen camera. We derive an imaging formation model for the under-screen camera, which facilitates the generation of screen-pixel perturbations on synthesized datasets. We design and implement a method to successfully fool different deep learning models. We believe this is a pioneer work which can stimulate a lot of follow-up works either to attack or safeguard current under-screen cameras on mobile devices.

Acknowledgements - We thank the anonymous reviewers and the shepherd for their valuable suggestions. This work has been funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement ENLIGHT'EM No. 814215.

REFERENCES

[1] Aug 2021. Xiaomi announces Mix 4 with under-display camera. *https://www.theverge.com/2021/8/10/22617979/xiaomi-mi-mix-4-announced-under-display-camera-cup-specs-price* (Aug 2021).

[2] July 2021. ZTE launches its new-generation under-display camera smartphone Axon 30. *https://www.zte.com.cn/global/about/news/20210727e1.html* (July 2021).

[3] Ams. 2019. AMS launches optical sensor which measures ambient light from behind a smartphone's OLED screen. (Jan. 2019). https://ams.com/-/ams-launches-optical-sensor-which-measures-ambient-light-from-behind-a-smartphone-s-oled-scre-1

[4] Nick Antipa, Grace Kuo, Reinhard Heckel, Ben Mildenhall, Emrah Bostan, Ren Ng, and Laura Waller. 2018. DiffuserCam: lensless single-exposure 3D imaging. *Optica* (2018), 1–9.

[5] M Salman Asif, Ali Ayremlou, Aswin Sankaranarayanan, Ashok Veeraraghavan, and Richard G Baraniuk. 2016. Flatcam: Thin, lensless cameras using coded aperture and computation. *IEEE Transactions on Computational Imaging* (2016), 384–397.

[6] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. 2018. VGGFace2: A Dataset for Recognising Faces across Pose and Age. In *Proceedings of the IEEE Automatic Face & Gesture Recognition (FG)*.

[7] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

[8] Sung-Ho Chae, Cheol-Hwan Yoo, Jee-Young Sun, Mun-Cheon Kang, and Sung-Jea Ko. 2017. Subpixel rendering for the pentile display based on the human visual system. *IEEE Transactions on Consumer Electronics* (2017), 401–409.

[9] Gary Charness, Uri Gneezy, and Michael A Kuhn. 2012. Experimental methods: Between-subject and within-subject design. *Journal of economic behavior & organization* (2012), 1–8.

[10] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Proceedings of International Conference on Machine Learning (ICML)*.

[11] Ambra Demontis, Marco Melis, Maura Pintor, Jagielski Matthew, Battista Biggio, Oprea Alina, Nita-Rotaru Cristina, and Fabio Roli. 2019. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *Proceedings of the USENIX Security Symposium (USENIX Security)*.

[12] Android developers. 2023. Canvas. *https://developer.android.com/reference/android/graphics/Canvas* (2023).

[13] Android developers. 2023. Multi-window support. *https://developer.android.com/guide/topics/large-screens/multi-window-support* (2023).

[14] Android developers. 2023. Notifications. *https://developer.android.com/design/ui/mobile/guides/home-screen/notifications* (2023).

[15] Android developers. 2023. Update other apps' media files. *https://developer.android.com/training/data-storage/shared/media* (2023).

[16] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[17] Ruicheng Feng, Chongyi Li, Huaijin Chen, Shuai Li, Jinwei Gu, and Chen Change Loy. 2023. Generating Aligned Pseudo-Supervision from Non-Aligned Data for Image Restoration in Under-Display Camera. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[18] Ruicheng Feng, Chongyi Li, Huaijin Chen, Shuai Li, Chen Change Loy, and Jinwei Gu. 2021. Removing diffraction image artifacts in under-display camera via dynamic skip connection network. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[19] KeMing Gao, Meng Chang, Kunjun Jiang, Yaxu Wang, Zhihai Xu, Huajun Feng, Qi Li, Zengxin Hu, and YueTing Chen. 2021. Image restoration for real-world under-display imaging. *Optics Express* (2021), 37820–37834.

[20] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[21] Joseph W Goodman and P Sutton. 1996. Introduction to Fourier optics. *Quantum and Semiclassical Optics-Journal of the European Optical Society Part B* (1996), 1095.

[22] Ke Gu, Min Liu, Guangtao Zhai, Xiaokang Yang, and Wenjun Zhang. 2015. Quality assessment considering viewing distance and image resolution. *IEEE Transactions on Broadcasting* 61, 3 (2015), 520–531.

[23] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2018. Countering Adversarial Images using Input Transformations. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[25] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. 2019. Searching for MobileNetV3. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[26] Stepan Komkov and Aleksandr Petiushko. 2021. AdvHat: Real-world adversarial attack on ArcFace face ID system. In *Proceedings of International Conference on Pattern Recognition (ICPR)*.

[27] George Alex Koulieris, Kaan Akşit, Michael Stengel, Rafał K Mantiuk, Katerina Mania, and Christian Richardt. 2019. Near-eye display and tracking technologies for virtual and augmented reality. In *Proceedings of Computer Graphics Forum*. Wiley Online Library.

[28] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security*. Chapman and Hall/CRC.

[29] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

[30] Yuezun Li, Xin Yang, Baoyuan Wu, and Siwei Lyu. 2019. Hiding faces in plain sight: Disrupting ai face synthesis with adversarial perturbations. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[31] Hao Liu, Hanting Ye, Jie Yang, and Qing Wang. 2021. Through-Screen Visible Light Sensing Empowered by Embedded Deep Learning. In *Proceedings of the ACM SenSys Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things (AIChallengeIoT)*.

[32] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. 2021. *SLAP*: Improving physical adversarial examples with $Short - Lived$ adversarial perturbations. In *Proceedings of the USENIX Security Symposium (USENIX Security)*.

[33] Xudong Lv and Z Jane Wang. 2009. Reduced-reference image quality assessment based on perceptual image hashing. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

[34] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[35] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[36] Apostolos Modas, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. 2019. Sparsefool: a few pixels make a big difference. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[37] Seyed-Mohsen Moosavi-Dezfoli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[38] Hong-Wei Ng and Stefan Winkler. 2014. A data-driven approach to cleaning large face datasets. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

[39] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. 2021. Bag of Tricks for Adversarial Training. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[40] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[41] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM Asia Conference on Computer and Communications Cecurity (ASIACCS)*.

[42] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE.

[43] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

[44] Scott Pudlewski and Tommaso Melodia. 2013. Compressive video streaming: Design and rate-energy-distortion analysis. *IEEE Transactions on Multimedia* (2013), 2072–2086.

[45] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. 2018. Certified Defenses against Adversarial Examples. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* (2015), 211–252.

[47] Graeme D Ruxton. 2006. The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. *Behavioral Ecology* (2006), 688–690.

[48] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[49] Athena Sayles, Ashish Hooda, Mohit Gupta, Rahul Chatterjee, and Earlence Fernandes. 2021. Invisible perturbations: Physical adversarial examples exploiting the rolling shutter effect. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[50] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[51] Hrishikesh Panikkasseril Sethumadhavan, Densen Puthussery, Melvin Kuriakose, and Jiji Charangatt Victor. 2020. Transform Domain Pyramidal Dilated Convolution Networks for Restoration of Under Display Camera Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[52] Ali Shahin Shamsabadi, Ricardo Sanchez-Matilla, and Andrea Cavallaro. 2020. ColorFool: Semantic adversarial colorization. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[53] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2020. Fawkes: Protecting privacy against unauthorized deep learning models. In *Proceedings of the USENIX Security Symposium (USENIX Security)*.

[54] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*.

[55] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. 2018. When does machine learning *FAIL*? generalized transferability for evasion and poisoning attacks. In *Proceedings of the USENIX Security Symposium (USENIX Security)*.

[56] Varun Sundar, Sumanth Hegde, Divya Kothandaraman, and Kaushik Mitra. 2020. Deep atrous guided filter for image restoration in under display cameras. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[57] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. 2017. Inception-v4, Inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*.

[58] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

[59] Richard Szeliski. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.

[60] Takatoshi Tsujimura. 2017. *OLED display fundamentals and applications*. John Wiley & Sons.

[61] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

[62] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. 2018. With great training comes great vulnerability: Practical attacks against transfer learning. In *Proceedings of the USENIX Security Symposium (USENIX Security)*.

[63] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* (2004), 600–612.

[64] Xingxing Wei, Ying Guo, and Jie Yu. 2022. Adversarial sticker: A stealthy attack method in the physical world. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), 2711–2725.

[65] Emily Wenger, Shawn Shan, Haitao Zheng, and Ben Y Zhao. 2023. SoK: Anti-Facial Recognition Technology. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

[66] wikipedia. 2022. Camfecting. *https://en.wikipedia.org/wiki/Camfecting* (December 2022).

[67] Anqi Yang and Aswin C Sankaranarayanan. 2021. Designing display pixel layouts for under-panel cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 2245–2256.

[68] Hanting Ye and Qing Wang. 2021. SpiderWeb: Enabling Through-Screen Visible Light Communication. In *Proceedings of the ACM Embedded Networked Sensor Systems (SenSys)*.

[69] Hanting Ye, Jie Xiong, and Qing Wang. 2023. When VLC Meets Under-Screen Camera. In *Proceedings of the ACM Mobile Systems, Applications, and Services (MobiSys)*.

[70] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. 2014. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923* (2014).

[71] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*.

[72] Xucong Zhang, Seonwook Park, Thabo Beeler, Derek Bradley, Siyu Tang, and Otmar Hilliges. 2020. ETH-XGaze: A large scale dataset

for gaze estimation under extreme head pose and gaze variation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[73] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. 2020. Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[74] Yuqian Zhou, Michael Kwan, Kyle Tolentino, Neil Emerton, Sehoon Lim, Tim Large, Lijiang Fu, Zhihong Pan, Baopu Li, Qirui Yang, et al. 2020. UDC 2020 challenge on image restoration of under-display camera: Methods and results. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[75] Yuqian Zhou, David Ren, Neil Emerton, Sehoon Lim, and Timothy Large. 2021. Image restoration for under-display camera. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

[76] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. Automating visual privacy protection using a smart LED. In *Proceedings of the ACM Mobile Computing and Networking (MobiCom)*.