Background and Occlusion Defenses Against Adversarial Examples and Adversarial Patches

Michael McCoyd

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-170
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-170.html

August 14, 2020

Acknowledgement

My advisor, David Wagner, has been a patient and insightful mentor who gave me many opportunities to explore and learn new things. Thank you for the chance to learn so much and for showing me how to see which are the important questions and which of them we might make progress on. And for the chance to help pass that on to a few others as well.

One of the best parts of this has been all of the collaborations on works that made it to publication and some that did not. Thank you.

Berkeley is a magical place. It has been a privilege to be here.

My parents get the deepest thanks for loving us and teaching us to wonder and question. I am very grateful that the notion of being a scientist was always part of our household. It has stuck with me.

(Abbreviated version)

Background and Occlusion Defenses Against Adversarial Examples and Adversarial Patches

by

Michael Joseph McCoyd

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David Wagner, Chair
Professor Anthony D. Joseph
Professor David Bamman

Summer 2020

Background and Occlusion Defenses Against Adversarial Examples and Adversarial Patches

Abstract

Background and Occlusion Defenses Against Adversarial Examples and Adversarial Patches

by

Michael Joseph McCoyd

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor David Wagner, Chair

Machine learning is increasingly used to make sense of our world in areas from spam detection, recommendation systems, to image classification. However, in each, it is vulnerable to adversarial manipulation. Within adversarial machine learning, we examine image classification attacks and defenses. We construct spoofs of face detection, and we create defenses against two attacks on image classification: normal and patch adversarial example attacks.

We examine the Viola-Jones 2D face detection algorithm to study whether images can be created that humans do not notice as faces, yet the algorithm detects as faces. We show that it is possible to construct images that Viola-Jones recognizes as containing faces, yet no human would consider a face. Moreover, we show that it is possible to construct images that fool facial detection even after the images are printed and then photographed.

Adversarial examples allow crafted attacks against deep neural network classification of images. The attack changes the computer classification of an image without changing how humans classify it.

We propose a defense of expanding the training set with a single, large, and diverse class of background images, striving to ‘fill’ around the borders of the classification boundary. We find that our defense aids the detection of simple attacks on EMNIST, but not advanced attacks. We discuss several limitations of our examination.

An attacker limited to changing just a small patch of an image can still deceive deep learning image classification. We propose a defense against such patch attacks based on multiple partial occlusions of the image such that a few occlusions each completely hide the patch. We provide certified accuracy for CIFAR-10, Fashion MNIST and MNIST, with a tunable tradeoff between the false-positive rate and certified accuracy. For CIFAR-10 and a 5×5 patch, we can provide certified accuracy for 43.8% of images, at the cost of only 1.6% in clean image accuracy compared to the architecture we defend or a cost of 0.1% compared to our training of that architecture, including a 0.2% false-positive rate.

Fiat lux,
non est mendacium.

To my parents,
my brother and sister,
and all my teachers.

Contents

List of Figures

List of Tables

Acknowledgments

Many people made the completion of this degree and work possible, some mentioned here.

My advisor, David Wagner, has been a patient and insightful mentor who gave me many opportunities to explore and learn new things. Thank you for the chance to learn so much and for showing me how to see which are the important questions and which of them we might make progress on. And for the chance to help pass that on to a few others as well.

My dissertation and qualifying exam committee members Anthony Joseph, David Bamman and Dawn Song gave me invaluable feedback to guide my research. Thank you.

One of the best parts of this has been all of the collaborations on works that made it to publication and some that did not. I learned from working with each of you, and I enjoyed our many discussions trying to figure things out. Thank you David Wagner, Petros Maniatis, Sanjit A. Seshia, Cynthia Sturton, Nicholas Carlini, Rohit Sinha, Thurston H.Y. Dang, Sakshi Jain, Wei Yang Tan, Won Park, Steven Chen, Neil Shah, Ryan Roggenkempler, Minjune Hwang, Jason Liu, Zhanyuan Zhang and Benson Yuan. It was fun and rewarding.

My colleagues in the Secure Computing group made Berkeley a great community experience. Thank you to my office mates Cynthia Sturton, Erika Chin, Thurston Dang, Richard Shin, David Fifield and Dan Hendrycks for a great environment and conversations. Also to the rest of our group for exposure to broader ideas and always interesting conversations next door or down the hall: Nicholas Carlini, Paul Pearce, Grant Ho, Chris Thompson, Kurt Thomas, Jethro Beekman, Brad Miller, Chawin Sitawarin, Austin Burdock, Nathan Malkin, Linda Lee, Pratyush Mishra, Sakshi Jain, Rishabh Poddar, Warren He and Frank Li.

Few of us would have made it without our fabulous department and group staff, particularly Angie Abbatecola, who kept the Secure Computing group well taken care of.

Berkeley itself is a magical place. It has been a privilege to be here.

I'm grateful for generous gifts of support from Google and Futurewei, the Hewlett Foundation through the Center for Long-term Cybersecurity and from Intel through the ISTC for Secure Computing. Without them, this would have been harder.

In getting to Berkeley, Calvin Lin, at UT Austin, played a key role in letting me join his freshman class and later his program, which started all this. His program and students provided a great home for me while I was there. Mike Dahlin had me join his project and then found something new for me to work on when projects shifted.

In our beautiful California, the Berkeley scientific diving program and Jim Hayward gave me a home outside of research. If I ever have a chance to help with oceanographic research as a diver, I have the core credentials. Monterey's inquisitive and playful sea lions and curious but more shy harbor seals provided many smiles and enjoyable dive buddies.

My sister and brother, Elizabeth and Matthew, have served as examples by pursuing their dreams and advanced degrees and have always made us a great trio. It is always great to have you there.

Our parents, Blanche Helen Flaherty McCoyd and Gerard Charles McCoyd, get the deepest thanks for loving us and teaching us to wonder and question. I am very grateful that the notion of being a scientist was always part of our household. It has stuck with me.

Chapter 1

Introduction

Machine learning systems, such as neural and deep convolutional networks, are increasingly used in decision or classification tasks. Their use includes: recommending songs or books, classifying whether emails are spam, routing shipments, detecting fraud in business and banking, recognizing spoken requests, identifying people and objects in scenes, and controlling robots and vehicles. In each of these areas, adversaries may compete for advantage. While machine learning systems are very capable, they are also vulnerable to being deliberately fooled, often with high success.

Adversarial machine learning examines how to use machine learning in a context where it is under attack by a knowledgeable and motivated adversary [HJN+11]. It looks at the machine architectures and methods used for the tasks above or more closely at particular uses. It examines why machine learning is vulnerable and tries to build more robust systems. That robustness might be against manipulation while it is learning, while it is updating to a changing world or while it is making judgments about that world.

Our work focuses on adversarial machine learning in the visual environment and against attacks during classification. Visual recognition provides one entry into examining the interplay of normal machine learning performance and the cycle of attack and defense. It has security concerns given the use of machine learning for such things as facial recognition access control, self-driving vehicles, and content classification.

In image recognition, one attack is to have the scene, or part of the scene, be something that looks to humans like one thing yet is classified by the computer as something else. These are called adversarial examples and attack the system when it is classifying at test time, not when it is learning [SZS+14, GSS15]. They typically are created from an existing image that humans would classify one way, such as containing a person, car, horse, or plane. The adversarial example makes small changes to the image or scene to cause the computer to classify it differently than the human. The change might be limited to very small adjustments of the pixel values throughout the image, or it might change only a very small area of the image. Depending on the magnitude of the changes or the size of the area changed, the change can go unnoticed by humans yet switch the machine classification.

We will describe three efforts we have made in this field. Our first work looked at fooling a common face detection system into thinking that a face is present when humans see none. From there, our second work looked at defending image classification against some known attacks that can make small pixel changes throughout the image. Finally, our most recent work looks at defending image classification against any possible attack that limits its change to a small area of the image. We provide a bit more introduction to these works below and describe them in detail in the following chapters.

1.1 Spoofing 2D Face Detection

In this work, we look at 2D face detection, which has many applications. For instance, face detection and face recognition have been used for user authentication, tagging social media photos, video surveillance, physical security, and other biometric security measures. Face detection determines whether a face is present in an image and possibly determines where the face is in the image. Face recognition classifies a face as being a certain person, not being them, or being in or not in a certain group of people, such as those authorized for access.

The security of facial recognition in the attended setting was not well-studied. An example of an unattended setting would be a system in isolation recognizing a specific person, such as to allow them access, such as a laptop allowing access to a person in an otherwise empty room. An attended setting would have humans in the area directly or casually related to the automated system.

An attended example would be a security guard at a building security gate with a few automated recognition lanes. An automated system does the recognition, but the guard prevents strange things from happening, such as holding a picture of the CEO up to the scanner. Or the system might include periodic video review.

Facial recognition could also be used for authenticating to a computer or end-user device (machine access control) where there were other employees present in the vicinity who might have an opportunity to notice attacks. The attended setting can also be relevant here. In either situation, holding up pictures of authorized individuals could raise alarms.

In some applications, it is the attended setting that is arguably more relevant. It is important to know whether there are attacks even a human wouldn't detect if they were present when the attack occurred.

In this paper, we study attacks on facial detection in the attended setting: we study whether it is possible to fool face detection algorithms into thinking an extra face is present, while preventing humans (e.g., security guards, others in the vicinity) from noticing the extra face. We do not address the later stage of face recognition, of who that face belongs to.

The most commonly used algorithm for facial detection, Viola-Jones[VJ01] §2.2, works by using machine learning techniques to build a classifier that determines whether a region of an image contains a face or not. We construct successful attacks on the Viola-Jones algorithm.

Our approach uses a feedback-guided search algorithm to construct an image that Viola-Jones recognizes as a face, yet is unlikely to be recognized by a human. We select a cover image C that does not contain a face; for instance, C might simply be an all-white image.

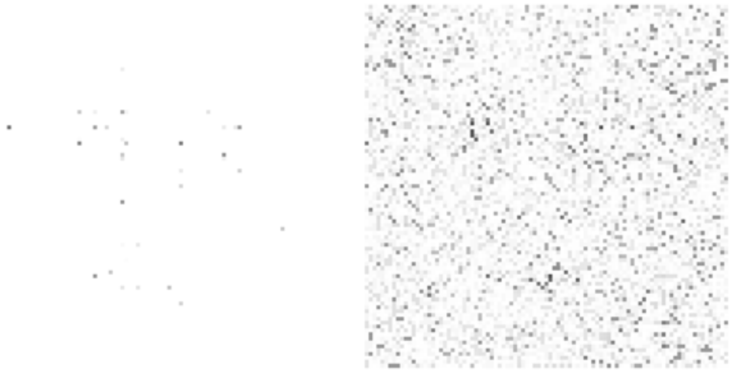


Figure 1.1: Two malicious images we generated with our algorithms. The left image is detected as a face if fed directly into a face detector. The right image is detected as a face, even after it is subjected to distortions imposed by the physical world: When displayed on a tablet and captured using a camera, the resulting image is detected as a face 97% of the time.

We start with an ordinary image of a face (recognized as a face by humans and Viola-Jones alike), F, and iteratively modify F. At each step in our algorithm, we make a small random modification to F to make it more similar to C, but while ensuring that F remains recognized as a face by Viola-Jones.

Essentially, our algorithm uses the Viola-Jones classifier to provide feedback and guide a directed random walk through the space of images, probing the decision boundary of the classifier to search for an image that is as similar to C as possible while still being classified as a face by Viola-Jones. Through appropriate instantiation of this approach, we are able to create digital spoof images that humans do not notice, yet Viola-Jones detects if they are presented directly to the facial detection algorithm with no modifications.

We then refine our algorithm to deal with degradation during delivery imposed by the physical world. We imagine conducting our attack in a simplified physical world, one where the image is presented squarely to the camera. We model this world with a simulated analog channel, §2.6. By modeling the effects observed in our experiments, we are able to create a reasonable simulation of the kinds of degradation imposed by the physical world and then adjust our attack (§2.8) to create spoof images that are more robustly detected despite degradation imposed by the physical world.

Our attacks are successful. See figure 1.1 for two examples of malicious images we generated using our techniques.

The contributions of this work are that we show new attacks on facial detection; introduce a new algorithm to construct inputs that fool a classifier, using binary outputs from the classifier; and devise techniques for dealing with noise and image degradation introduced by the physical world.

In our work, we proposed that if our results could be extended to spoofing facial recognition, an attacker might be able to display a spoof image as the cover of a notebook and casually hold it in view of a security camera, thereby gaining access to a protected computer or physical location. After we did this work, Sharif et al. [SBBR16] showed that it is possible to spoof face recognition with custom 3D printed glasses frames.

1.2 Background Class Defense Against Adversarial Examples

Deep neural networks have been very successful at many classification tasks. Yet they have been found to be vulnerable to misclassifying deliberately crafted images [SZS$^+$14, GSS15].

We examine whether adding a large and diverse background class to the training data can help detect and neutralize adversarial examples. We assume that there is a set of key classes that we care about distinguishing between. We propose to train the classifier by adding additional training images that are not from the key classes that we care about distinguishing between. We introduce a new class that we call the background class for these additional images. The background class effectively serves as a "none of the above." For example, if we want to recognize MNIST digits, we might use images of non-digit handwriting as examples of the background class. The background class is intended to create a default classification to provide better separation between the key classes within the model's feature space. We measure how this defense affects the classification of normal images and the detection of adversarial examples.

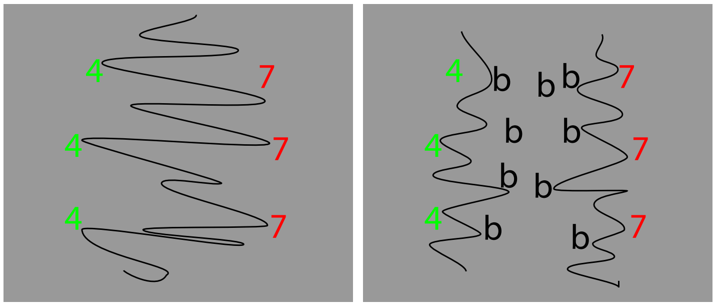
Figure 1.2 illustrates a simplified visualization of the decision boundary of an undefended classifier (left) and our proposed defense (right). We hope that the background class makes it harder to modify an image from one key class to be misclassified as another key class.

These background images are not noise, but images of objects distinct enough from our key classes to be distinguished from them by the model.

In use, our network is still evaluated on its ability to distinguish between images that humans perceive as key classes. We change the classification task from image → one of key classes to image → one of key classes or background.

We evaluate this idea with the EMNIST dataset [CATv17], of handwritten digits and letters, and the fast gradient sign [GSS15], fast gradient [LCLS17], and Carlini [CW17a] attacks.

We find our defense aids detection of simple attacks, but we have not found any increase in detection of state of the art attacks. There are weaknesses in our examination; in particular, the letters of our background class are not crafted for the role of separating the digits in image or feature space.



(a) The original classifier (b) With background class (our defense)

Figure 1.2: The intuition behind our defense. In (b), we introduce an extra class, the background class, for images other than the classes we want to recognize. The hope is that this separates those classes enough that adversarial examples are no longer effective.

1.3 Minority Reports Defense Against Adversarial Patch

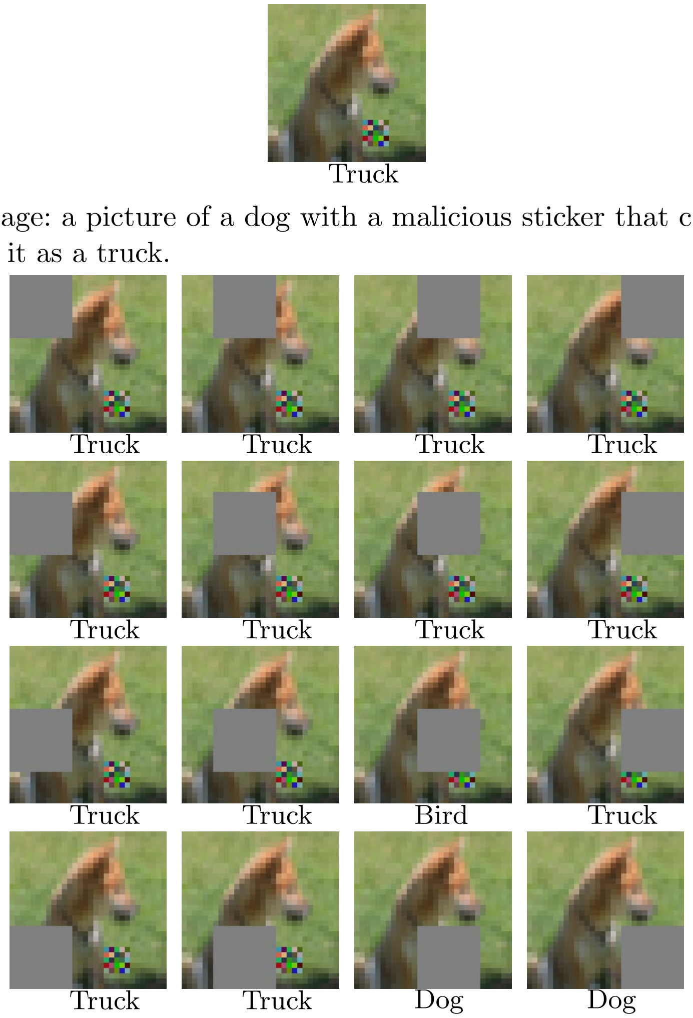
Recently, researchers have proposed the adversarial patch attack [BMR$^+$17, KZG18], where the attacker changes just a limited rectangular region of the image, for example, by placing a sticker over a road sign or adding a patch to a scene. In this work, we propose a defense against this attack.[1] For simplicity, we restrict the attacker to a square patch.

The idea of our defense is to occlude part of the image and then classify the occluded image. If we knew the location of the patch, we could occlude that region of the image (e.g., overwriting it with a uniform grey rectangle), then apply the classifier to the occluded image. If we can train a classifier that properly classifies occluded images, this would defend against such an attack, as the attacker's contribution is completely overwritten and the input to the classifier (the occluded image) cannot be affected by the attacker in any way.

In practice, we do not know the location of the adversarial patch, so a more sophisticated defense is needed. Our approach works by enumerating all possible locations for the occluded region, classifying each resulting occluded image, and then analyzing the classifier's predictions on these occluded images. If the occlusion region is larger than the adversarial patch, we show that it is possible to detect many attacks. The idea is that if the occlusion is larger than the adversarial patch, several of the occluded images will completely obscure the adversarial patch, and thus the classifier's prediction on those images will be unaffected by the adversary and will hopefully match the correct label. We show how to use this redundancy to detect adversarial patch attacks. We call our scheme the minority reports defense[2] because there will

[1] Minority Reports Defense: Defending Against Adversarial Patches. Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Liu, and David Wagner. SiMLA 2020.

[2] Steven Spielberg. Minority Report. 20th Century Fox, 2002.



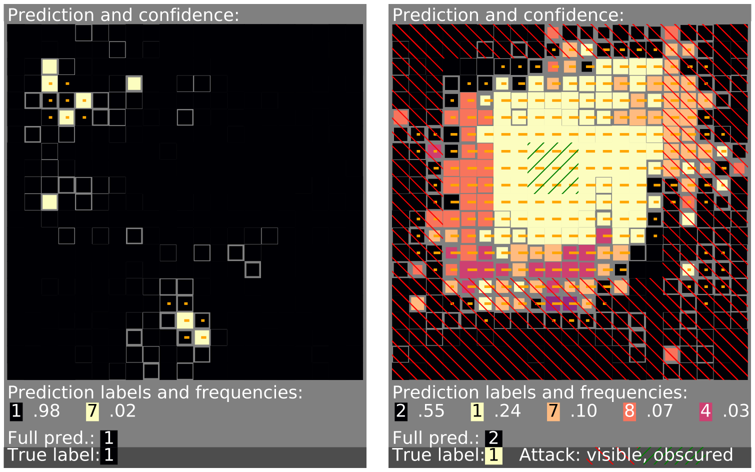
(a) An attack image: a picture of a dog with a malicious sticker that causes a standard model to classify it as a truck.

(b) We occlude part of the image with a grey square, then classify these occluded images. Here the 3rd and 4th predictions in the 4th row will be unaffected by this attack. Our actual defense ensures that any attack will be fully occluded by a 3 × 3 grid of predictions, instead of the 1 × 2 grid shown here.

Figure 1.3: Our scheme works by occluding different portions of the image and analyzing the predictions made by the classifier on these occluded images.

always be a minority of predictions that cannot be influenced by the attacker and (presumably) vote for the correct label.

Figure 1.3 illustrates how we turn one input image, Figure 1.3a, into a grid of partially occluded images, Figure 1.3b, such that any attack will be occluded in a cluster of several predictions.We then apply the classifier to each occluded image to obtain a grid of predictions. We can see that there will always be a cluster that fully obscures the attack and thus whose



(a) Benign (b) Attack

Figure 1.4: Prediction grids for a benign image (left) and an attack image (right). Each cell in the grid is colored based on the prediction made by the classifier when fed an image obscured at that position in the grid. A cluster of identical minority predictions, as seen in the right image, suggests an attack. In the attack image on the right, green hashes mark the nine predictions where the adversarial patch was fully occluded.

labels are expected to agree. Such a cluster that votes against the majority with high enough prediction confidences suggests an attack; see Figure 1.4.

We evaluate the certified accuracy of our scheme on CIFAR-10 [KH09], Fashion MNIST [XRV17], and MNIST [LBBH98], §4.6.

For CIFAR-10, we can prove that 44% of images are certified accurate (no matter where a sticker is placed, the resulting image will either be classified correctly or the attack will be detected) for 5 × 5 stickers, if we are prepared to accept that 0.2% of benign images are falsely treated as malicious. We are aware that these false positive rates may not be suitable for large-scale use. These CIFAR-10 results are with an inner model with rather low accuracy, 94%, not nearly state of art accuracy. We expect these results would be better with a more state of the art inner model.

In concurrent work, Chiang et al. study certified security against patch attacks using interval bounds propagation [CNA$^+$20]. They provide CIFAR-10 certify accuracy of 25% for a 5 × 5 sticker with a clean accuracy of 34% or of 30% with a clean accuracy of 48%, and certify accuracy of 42% for a 2 × 2 sticker with a clean accuracy of 92.9%.

For MNIST, if we treat a false positive the same as a misclassification, our defense can certify 64% of images accurate against 5 × 5 patches, with a model that achieves clean accuracy of 99.4% for MNIST (99.6% − 0.2% false positives). On MNIST, Chiang et al. can certify 62% of images safe against 5 × 5 patches, with a model that achieves clean accuracy of 92.9%.

Chapter 2

Spoofing 2D Face Detection

2.1 Introduction

Machine learning is increasingly used to make sense of data from sensors in the environment and for automated decision-making. In this work,[1] we look at 2D face detection. For instance, face detection and face recognition have been used for user authentication, tagging social media photos, video surveillance, physical security, and other biometric security measures.

Similar to other biometrics, the security of 2D face detection and recognition depends on whether it is used in attended or unattended settings. For example, a door thumbprint reader in an empty corridor is vulnerable to attacks that would not work in front of a guard at an entry gate. Attacks on unattended use of facial biometrics have been well studied from changing image perspective [DM09] to using a 2D picture of an authorized person with cutouts for the eyes[BCF$^+$13]. However, these attacks might not be effective if attempted in front of guards or even casual bystanders, as such an attack is noticeable and easily detected.

The security of facial recognition in the attended setting has not been well-studied. In some applications, it is the attended setting that is arguably more relevant. For instance, facial detection and recognition might be used for physical security and area access control; because deployments might include the presence of a guard or periodic video review, it is important to know whether there are attacks even a human wouldn't detect. Facial recognition could also be used for authenticating to a computer or end-user device (machine access control); because there might be other employees present in the vicinity who might have an opportunity to notice attacks, the attended setting is relevant here as well. In either situation, holding up pictures of authorized individuals could raise alarms. In this paper, we study attacks on facial detection in the attended setting: we study whether it is possible to fool face detection algorithms into thinking an extra face is present, while preventing humans (e.g., security guards, others in the vicinity) from noticing the extra face.

Why study facial detection, rather than facial recognition? Ultimately, it is the ability

[1] "Spoofing 2D Face Detection: Machines See People Who Aren't There", M. McCoyd and D. Wagner, arXiv:1608.02128, 2016

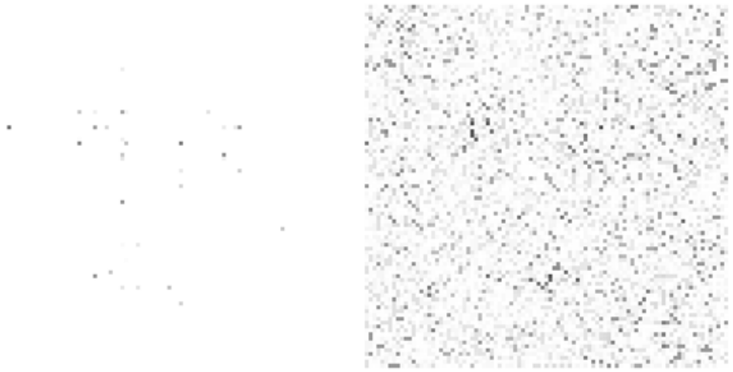


Figure 2.1: Two malicious images we generated with our algorithms. The left image is detected as a face if fed directly into a face detector. The right image is detected as a face, even after it is subjected to distortions imposed by the physical world: When displayed on a tablet and captured using a camera, the resulting image is detected as a face 97% of the time.

to fool facial recognition that matters. However, facial detection algorithms are simpler to study. In this paper, we focus on the security properties of facial detection. We view this as a first step towards the longer-term goal of analyzing facial recognition. Also, because facial recognition algorithms typically begin by first using facial detection to look for faces, then apply facial recognition to each detected face, any attack on facial recognition must begin by first fooling the facial detection step. Thus, we believe our techniques may have lasting relevance.

The most commonly used algorithm for facial detection, Viola-Jones[VJ01] §2.2, uses machine learning techniques to build a classifier that determines whether a region of an image contains a face or not. We construct successful attacks on the Viola-Jones algorithm. One technical challenge is that the facial detection algorithm only outputs a binary signal: face or non-face. Thus we can not use normal techniques for fooling classifiers, such as attempting to solve for a solution or using gradient descent to search for inputs that fool the classifier. Thus, we are forced to devise novel algorithms for constructing images that will fool the Viola-Jones classifier.

A second challenge is that the attacker cannot perfectly control the input to the classifier. Unlike spam, the adversary does not directly control the input to the classifier. Rather, the image passes through a noisy physical channel – the adversary can display one image, but the image captured by a camera will be reproduced only imperfectly. The signal is degraded by blurring, random noise, and other effects. We devise attacks that are robust to these effects.

Our approach uses a feedback-guided search algorithm to construct an image that Viola-Jones recognizes as a face, yet is unlikely to be recognized by a human. We select a cover image C that does not contain a face; for instance, C might simply be an all-white image. We start with an ordinary image of a face (recognized as a face by humans and Viola-Jones alike), F, and iteratively modify F. At each step we make a small random modification to F to make it more similar to C, but while ensuring that F remains recognized as a face by Viola-Jones.

Essentially, our algorithm uses the Viola-Jones classifier to provide feedback and guide a directed random walk through the space of images, probing the decision boundary of the classifier to search for an image that is as similar to C as possible while still being classified as a face by Viola-Jones. Through appropriate instantiation of this approach, we are able to create digital spoof images that humans do not notice yet Viola-Jones detects if they are presented directly to the facial detection algorithm with no modifications.

We then refine our algorithm to deal with degradation during delivery imposed by the physical world. We imagine conducting our attack in a simplified physical world, which we model with a simulated analog channel, §2.6. By modeling the effects observed in our experiments, we are able to create a reasonable simulation of the kinds of degradation imposed by the physical world and then adjust our attack (§2.8) to create spoof images that are more robustly detected despite degradation imposed by the physical world.

Our attacks are successful. See figure 2.1 for two examples of malicious images we generated using our techniques.

The contributions of this work are that we show new attacks on facial detection; introduce a new algorithm to construct inputs that fool a classifier, using binary outputs from the classifier; and devise techniques for dealing with noise and image degradation introduced by the physical world.

If it is possible to extend our results to spoof facial recognition as well, this might enable new, stealthier attacks on facial recognition: e.g., an attacker might be able to display a spoof image loop as the lock screen of a tablet and casually hold it in view of a security camera, thereby gaining access to a protected computer or physical location. We leave analysis of this threat to future work.

2.2 Background

The industry standard for face detection is the Viola-Jones classifier[VJ01]. It accepts a grayscale image and produces as output a boolean value, indicating whether the image is a face or not. Typically, to detect all faces in an image, we run the Viola-Jones classifier over all regions of the image and see which ones it classifies as a face.

For completeness, we provide a concise overview of Viola-Jones classifier, but for purposes of this paper, it is not important to understand the details of how the Viola-Jones classifier works. It is a boosted cascade classifier built out of multiple weak classifiers and trained on a set of face and non-face images. Each weak classifier computes the average intensity within two rectangles, subtracts these two numbers, and compares the result to a threshold to decide whether that portion of the image might contain a face (see figure 2.2). Early stages of the

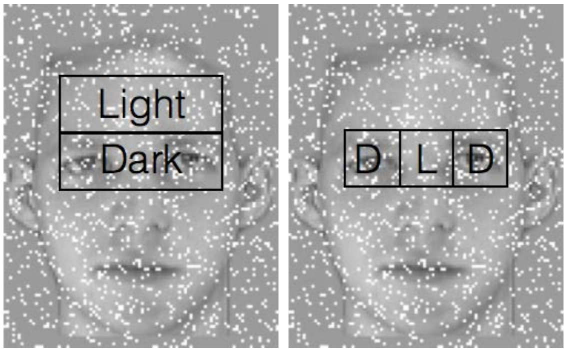


Figure 2.2: Two of Viola-Jones' early weak classifiers.

cascade use only a few weak classifiers with large rectangles; later stages use more classifiers with smaller rectangles. As a result, the Viola-Jones algorithm is very fast.

2.3 Problem Statement

Our attack model in this work is that the adversary might carry anything stealthy that lets them display an image crafted to fool facial detection. For example, one possible attack might be to carry a notebook with a printed image of a spoof image while approaching a security access point. The attacker could casually hold the notebook against his/her chest and neck as if they had just consulted some notes but were now finished with the notes, as they walk through the access point. Our attacks aim to cause the automated security system to notice one extra person. Ultimately, the goal would be to extend our attacks to fool facial recognition as well, so the system sees an authorized person who is not actually there and allows the attacker through — though we have not studied facial recognition, so our work should be viewed as only a first glimpse at what ultimately might be possible.

Many variants of this attack scenario are possible. Instead of a printed image, the attacker could carry a flat-panel display as part of the cover of a notebook, which might allow finer control over the displayed image — this is the scenario we focus on in this paper. One could also imagine an attacker who seeks to gain access to a computer that uses facial-recognition-based login (rather than password-based login). An attacker might find a publicly available image of the face of an authorized user, then use our techniques to try to disguise that image.

2.4 Starter Attacks

Our attack starts with an ordinary image of a face and morphs it until it is no longer recognizable to humans as a face. Roughly speaking, we do this by blending in enough of

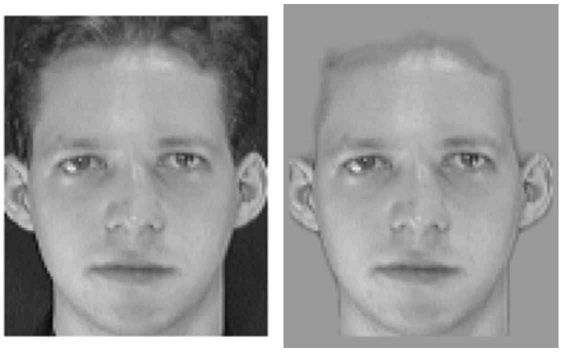


Figure 2.3: An example face from the AT&T database (left) and adjusted with flesh-toned background (right).

a cover image so that the original face is no longer detectable to humans while preserving enough of the original face so that a face detection algorithm[2] still detects the face.

Data To illustrate our attacks, we use faces from the AT&T Database of Faces[SH94]. Each face is an 8-bit 112 × 92 grayscale image with a dark background. Before applying our attacks, we use Photoshop's magic wand tool to replace the dark background and hair with a background with the same tone as the face. We also make the images 120 pixels tall by 96 wide by adding extra border pixels, to provide a bit more robustness to cropping. See figure 2.3 for an example.

Starter Attack: Blend

A very simple attack would be to construct the spoof image as a blend of the face and cover image. Each pixel, p, receives a fixed percent, r, of the cover value, with the remaining percentage from the face value. Specifically,

$$\text{spoof}_r(p) = r \times \text{cover}(p) + (1 - r) \times \text{face}(p), \tag{2.1}$$

where $r \in [0, 1]$ is constant for all p. We increase r until spoof_r is no longer detected by Viola-Jones. Figure 2.4 illustrates the attack with a cover image of granite.

This simple attack is unsuccessful. With a cover image of granite, Viola-Jones detects a face only for values of r in the range 0–0.16; humans can recognize a face for any value of r in the range 0–0.58, so there is no value of r which is accepted by Viola-Jones but not detected by humans. See figure 2.4 for an illustration.

We tested three cover images, and the attack fails for all three. With cover images of granite, sand, and all gray, values of r above 16%, 8%, and 45% fail Viola-Jones detection. Much

[2] Viola-Jones as implemented by OpenCV v3.1.1 using the haarcascade_frontalface_default.xml template file with cv2.CascadeClassifier.detectMultiScale called with default parameters plus CASCADE_SCALE_IMAGE

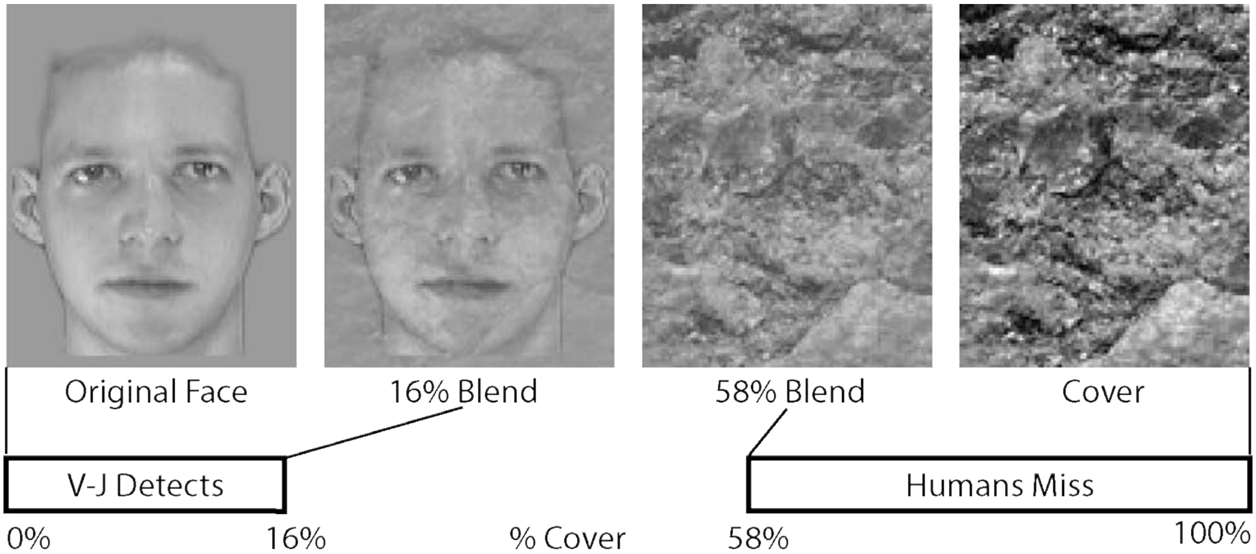


Figure 2.4: Blend attack showing the original face, two blends, and the cover image. The blends are the maximum percent of cover image that can be blended in while ensuring Viola-Jones still detects a face and the minimum percent that yields something not recognized as a face by humans. The attack fails as no blend percentages meet both conditions.

higher values are needed to start deceiving the human authors' detection, $58\%, 74\%,$ and 95%, respectively. There is no overlap between the images where humans don't see a face and images where Viola-Jones detects a face.

Starter Attack: Random Subset of Pixels

We also tried a different strategy: instead of blending all pixels, pick a random subset of pixels and replace them entirely with the corresponding pixel from the cover image. One might hope this will preserve the region differences used by Viola-Jones yet introduce enough detailed noise to fool the human. Specifically, for a given face, cover image, and fraction r, we choose

$$R = \text{ random set of } r \text{ fraction of the pixels} \tag{2.2}$$

and then define the spoof image as

$$\text{spoof}_r(p) = \begin{cases} \text{cover}(p) & \text{if } p \in R \\ \text{face}(p) & \text{otherwise} \end{cases} \tag{2.3}$$

To test this approach, we created 300 random images for each choice of four possible cover images and r in the range $[0.00, 1.00]$ in steps of 0.01. Cover images of granite, sand, all-gray, and all-white were used. The authors viewed five spoofs for each r, starting from $r = 1.00$, and judged when a face is first human-recognizable.

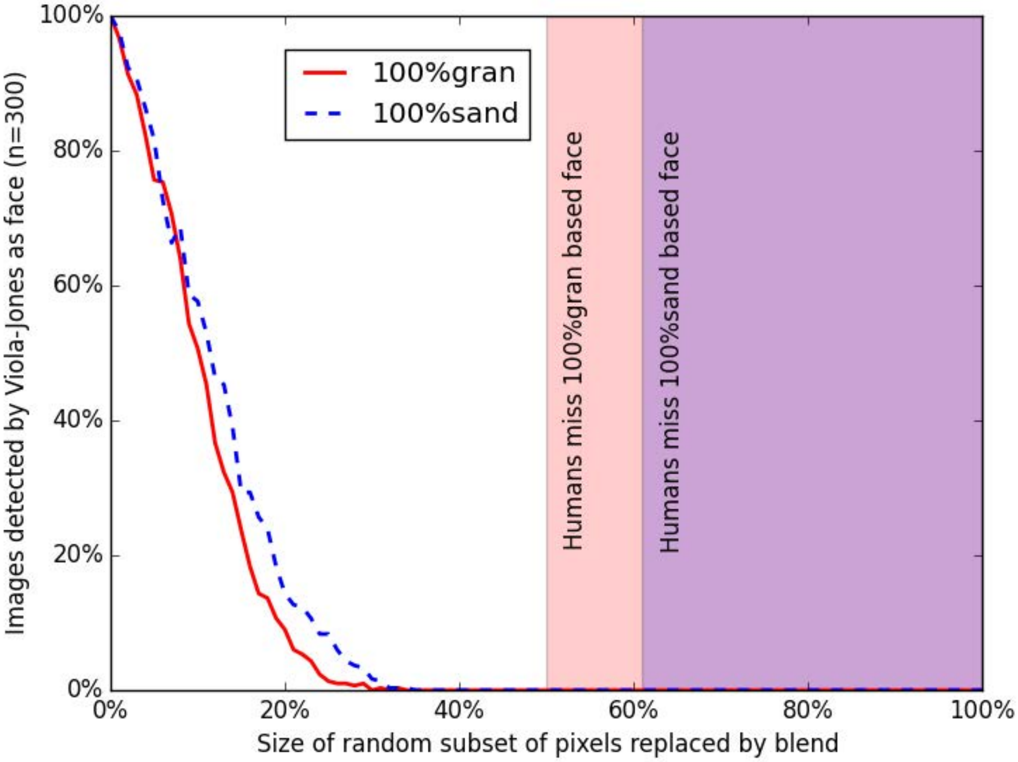


Figure 2.5: Random subset of pixels attack. The y-axis shows the fraction of candidate spoof images that were detected by Viola-Jones as a face. There are no images where the face is detected by Viola-Jones, yet missed by the humans. Only images built from granite or sand hid the face from the humans.

This attack also is unsuccessful. Only the granite and sand cover images had results that were at all promising. However, as Figure 2.5 shows, no spoofs were simultaneously detected as faces by Viola-Jones and missed by the humans. Thus this attack fails. But in the tail of the graph, there are images that Viola-Jones detected with $r = 29\%$, which is more than we saw in the blend attack, where $r = 16\%$ was the maximum achievable.

Starter Attack: Random of Blend

Another natural idea is to combine the previous two attacks. A slightly better attack might replace a percent, r, of random pixels from the original face with some blend of the face and cover image. For example, using the images of figure 2.4, we could replace some percent of random pixels in the original face with the pixels from the 58% blend of granite. Specifically, for a given face, cover image, blend rate $b \in [0, 1]$ and fraction of blend pixels $r \in [0, 1]$, the spoof image is defined as

$$\text{blend}_b(p) = b \times \text{cover}(p) + (1 - b) \times \text{face}(p), \tag{2.4}$$

$$R = \text{random set of } r \text{ fraction of pixels.} \tag{2.5}$$

$$\text{spoof}_r(p) = \begin{cases} \text{blend}_b(p) & \text{if } p \in R \\ \text{face}(p) & \text{otherwise} \end{cases} \tag{2.6}$$

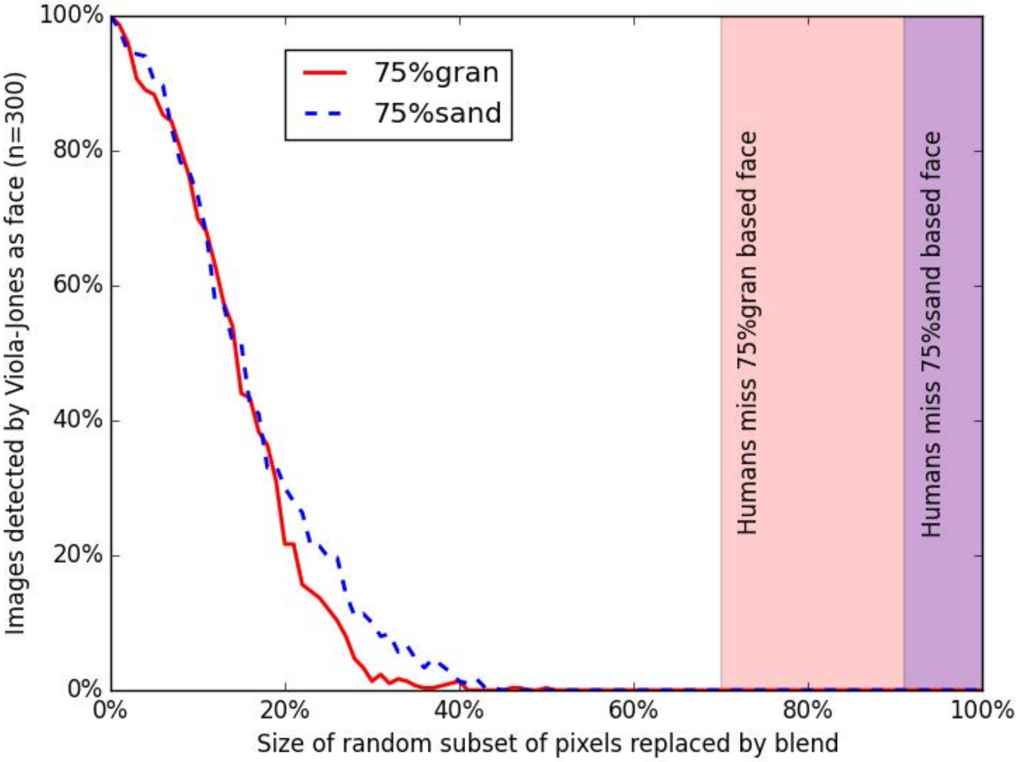


Figure 2.6: Random single blend attack: Detection rate of faces that replace a random set of pixels with a blend of the face and cover images. There are no images where the face is detected by Viola-Jones, yet missed by the humans. Only images built from 75% blends of granite or sand hid the face from the humans. All other images, including based on blends of gray or white, failed to hide the face from the humans.

We evaluated this attack in the same manner as the random choice attack, except that for each cover image, we created three blends of 25%, 50%, and 75% cover. A random 300 images were created for each combination of the cover image, blend and value of r.

Our results, in figure 2.6, show that we again can not satisfy Viola-Jones while fooling humans. Viola-Jones detection rates drop as the blend rate and random set size increase, and only two of the blends create a small range of images that hide the face from the authors. However, the bottom tails of the curves continue to show there are some series of choices of pixels that continue to succeed even after most other series of choices of pixels long ago failed. This suggests that perhaps further improvement might be possible by adding some guidance from Viola-Jones about the shift amount and/or choice of pixels.

2.5 Attack: Random Shift (Exact)

We build on these ideas to construct a more sophisticated attack. The attack applies small random changes to the image but reverts any changes that cause Viola-Jones to fail to detect a face. From our starter attacks, we learned that changing pixels only partway to the cover image allows us to retain Viola-Jones detection longer and that some sets of pixel choices work better than others. We combine these lessons by shifting random pixels small amounts toward the cover image and undoing those changes that cause a failure.

Our attack procedure has two parts: a search routine picks a suitable attack image while an oracle evaluates that attack image. The search is very simple. We have a loop that: picks a random pixel, changes its intensity halfway closer to the intensity of the corresponding pixel in the cover image, but rejects the change if the oracle says the face is no longer detected. The oracle checks whether a face would be detected in the spoof image at the same location as it is detected in the original face image. We first develop our attack, algorithm 1, with an oracle that passes the attack image directly to Viola-Jones, without accounting for any degradation that might be caused by the physical channel.

Experiment In preparing the data, we replace the area around the face with the corresponding pixels from the cover image (see figure 2.7). Much of the image background is not used by Viola-Jones, when detecting the face in the center of the attack image, so we use Viola-Jones to detect the location of this central face and preprocess our starting face to retain just the face area pixels.

Results This attack is successful. Because of our algorithm, all the results of our exact algorithm pass Viola-Jones detection. Typical results can be seen in the top row of figure 2.8, for cover images of granite, sand, all-white, and all-gray. The granite, sand and white attack images do not seem to stand out as faces to the authors, with the white attack image being particularly nice. In the other spoofs, we can see a grid pattern in the face region, presumably an artifact of the Viola-Jones regions.

Discussion The attack images in the results are successful in terms of creating images detected by Viola-Jones as a face but not noticed by humans as faces. Yet there is a problem. They are not detected as faces if we display these attack images on a retina resolution tablet, view them through a webcam as in figure 2.9, and pass the resulting webcam capture to Viola-Jones. The changes present in the resulting images can be seen in the bottom row of

Algorithm 1: Exact attack

```
Function Search($F$, $C$):
    Input  : Face image $F$,
             Cover image $C$
    Output : Spoof image $S$
    $S \leftarrow F$;
    while not Stalled($S$) do
        $p' \leftarrow$ a random pixel in $S$
        $T(p) \leftarrow \begin{cases} \text{Round}\left(\frac{S(p)+C(p)}{2}\right) & \text{if } p = p' \\ S(p) & \text{otherwise} \end{cases}$
        if Oracle($T$) then
            $S \leftarrow T$
    return S
Function Stalled($S$):
    Input  : Spoof Image $S$
    Output : Boolean
    Return True if $S$ has not changed in last several calls.
Function Oracle($T$):
    Input  : Test Image $T$
    Output : Boolean
    Return true if Viola-Jones detects a face in $T$ with bounds within 10% of those of
      the face in $F$.
```

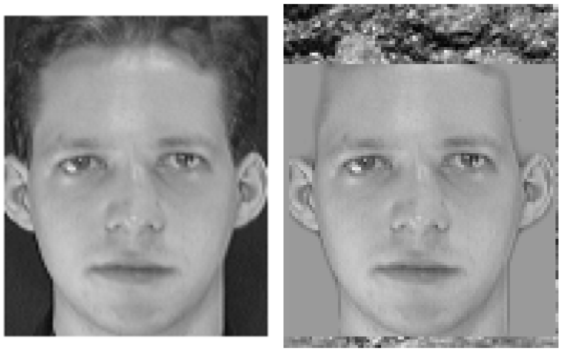


Figure 2.7: The face we used for the exact attack: the original AT&T image (left) and after altering the background and pasting the cover image around the face area (right).

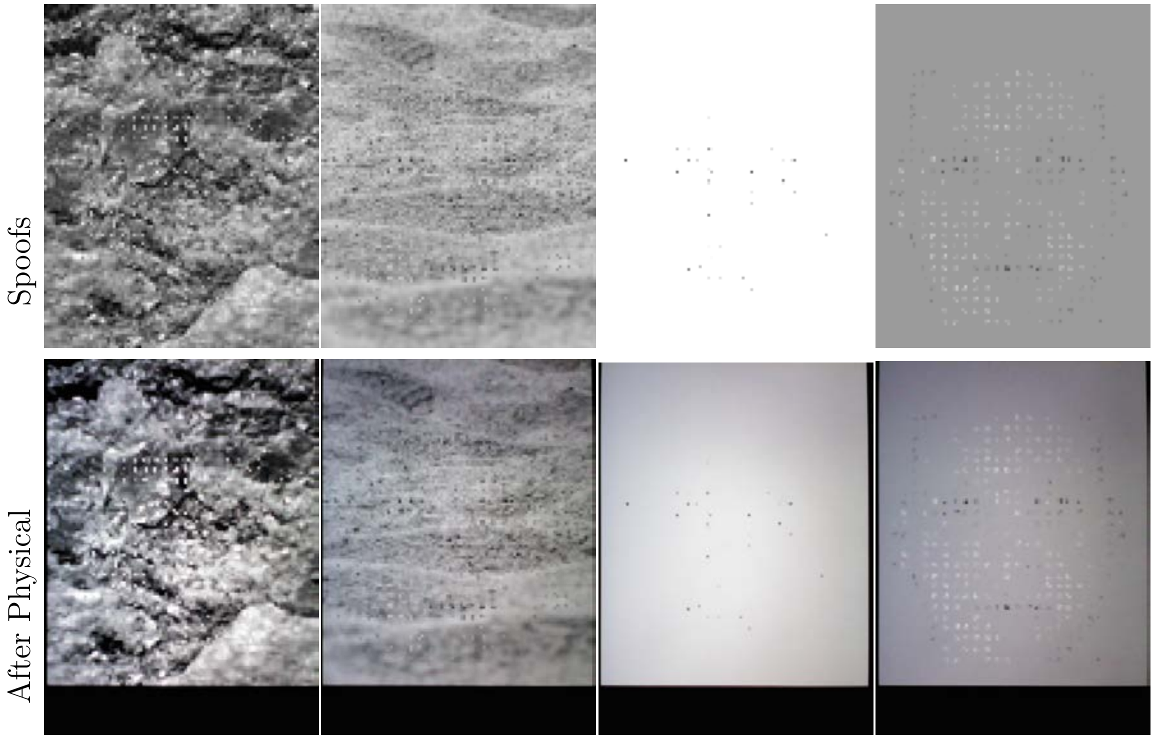


Figure 2.8: Results of the exact attack. The spoof images are detected as faces if fed directly into Viola-Jones. However, they are not detected by Viola-Jones after passing through the physical world.

figure 2.8. The failure of our spoof images in the physical world motivates the next steps of our attack.

2.6 Physical and Analog Channels

To extend our attack to the physical world, we created a simplified physical world and a simulation of it, an analog channel. We will use the term *physical channel* for passing an image through our display and webcam in a box, and the term *analog channel* for our software simulation of that.

Physical Channel To build a reproducible physical attack environment, we display the image on a tablet with a retina display in a low glare box with a fixed webcam. See figure 2.9. To reduce the effect of browser image smoothing, we expand the image several times before display, resulting in image pixels displayed as crisp squares. Apart from the fact that the display position is fixed, this is a fairly realistic simulation of an attack scenario, as the

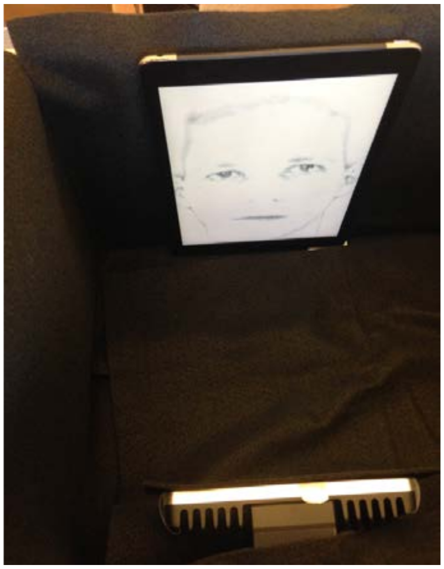


Figure 2.9: Our simplified physical world has a tablet with retina and a fixed webcam in a low-glare box.

attacker can carry a no-glare tablet and security webcams typically have fixed locations. We then measured the effect of the physical channel on images.

Individual Effects We found that passing images through our physical world has seven effects. We model five of them: brightening the image center, adding noise, adding Gaussian blur, reducing dark contrasts, and replicating pixels. We do not model barrel distortion, as we assume its effect is slight. We also do not yet model differences in alignment between the image and camera pixel borders. We created test images to help us measure each of these effects. As we consider each effect, in each figure, we show on the left a test image, in the center an image after the physical channel, and on the right an image after our analog channel.

To measure center brightening, we displayed a uniformly gray test image and captured 300 video frames of it. Averaging these frames gives us the average intensity for each pixel, figure 2.10, that the camera sees when this uniform gray image is displayed. We simulate this effect by adding to the input image the difference between the average image and the average image's average pixel value. Our analog channel winds up a bit darker than the physical channel.

The channel noise can be seen by first removing the center brightening effect. We subtract the average image from a single frame and then add back the test image. A detail of the result can be seen in figure 2.11. We get a noise distribution by subtracting from each frame the average image and estimate the distribution of the resulting differences (across all pixels). We found that this distribution is well-fit by a normal distribution with mean zero and a

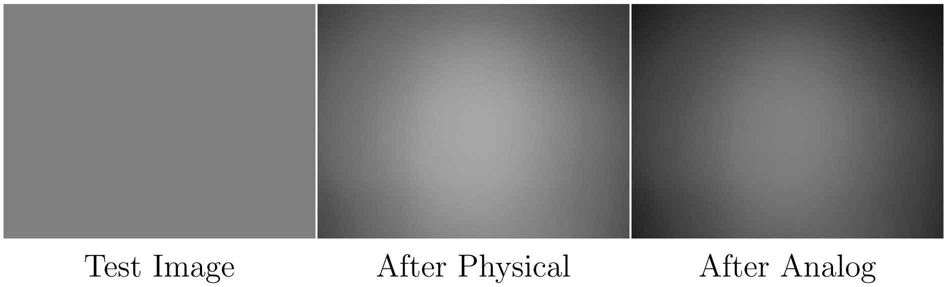


Figure 2.10: Channel center brightening.

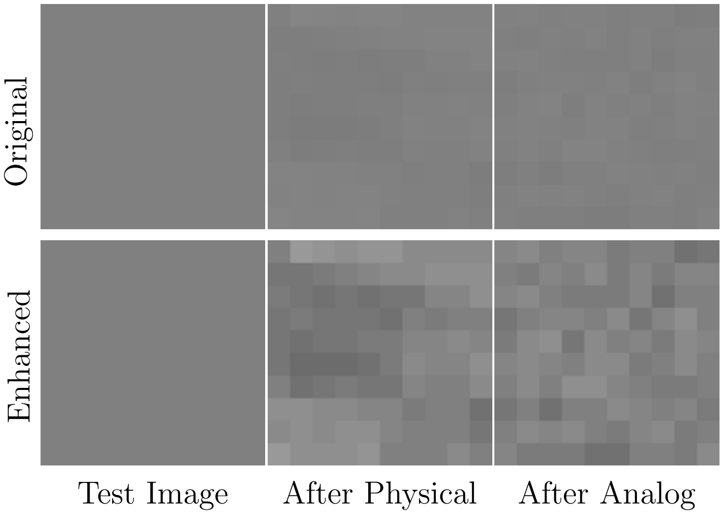


Figure 2.11: An example of channel noise seen in a 10 by 10 pixel detail of a gray image. The bottom row enhances the intensity differences by five.

standard deviation of 1.5 (see figure 2.12). Thus, to approximate the noise, our analog channel adds Gaussian noise with a standard deviation of 1.5 to the intensity of each pixel.

To assess channel blur, we displayed a test image with vertical bars whose width is one to four pixels. The result can be seen in figure 2.13. Our analog channel uses a Gaussian blur with a standard deviation of 0.9, as we found that this matches the observed effects well.

The physical channel has a non-linear effect on pixel intensities: for example, it reduces the contrast in dark regions and increases the contrast for light intensities, as shown in figure 2.14. We found that this can be modeled with a piecewise-linear intensity response curve applied uniformly to all pixels: a pixel with intensity x becomes intensity $f(x)$, where f is a piecewise-linear function. We found that two pieces are sufficient to fit the observed response curve well.

Our physical channel setup displays each 120-pixel-tall spoof image so that the image will fill most of the 480-pixel webcam height. As a result, each pixel of the spoof image fills

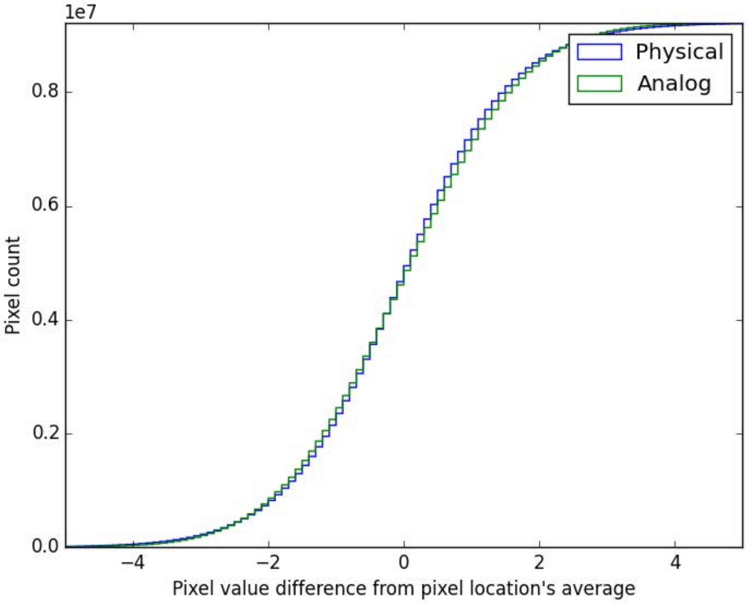


Figure 2.12: The distribution of additive noise imposed by the channel to each pixel intensity: We show both the observed distribution from the physical channel and the distribution of noise added by our simulated analog channel.

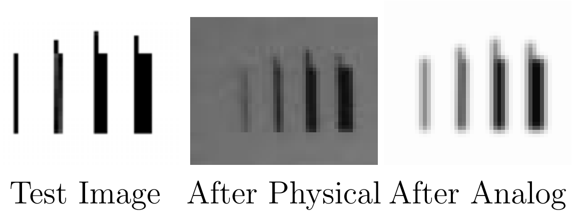
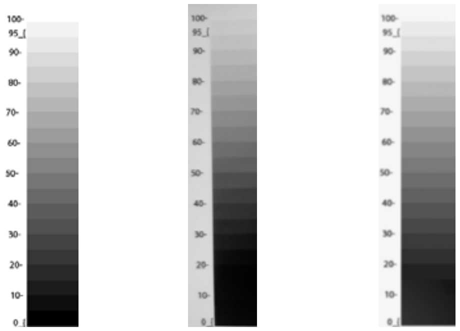


Figure 2.13: Channel blur effects.



Test Image After Physical After Analog

Figure 2.14: Channel contrast changes. Contrast is lost between darker values and gained between lighter ones.

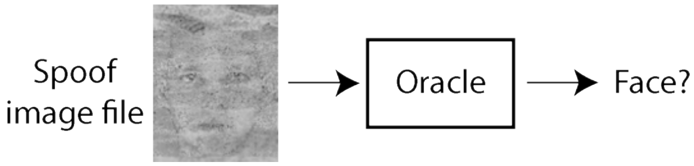


Figure 2.15: A naive oracle for our search algorithm.

approximately a 4x4 grid of pixels in the image captured by the webcam, though we make no attempt to position the spoof image to align exactly with a grid of four by four webcam pixels and the displayed image does not fill the webcam height exactly. Our analog channel simulates this by upscaling the 120x96 spoof image to a 480x384 image (using pixel replication, i.e., each pixel is replicated in a 4x4 grid) before applying the other effects.

Analog Channel Our analog channel simply chains together these individual effects: we apply upscaling, center brightening, response curve (contrast reduction), Gaussian blur, and Gaussian noise, in that order. Though our analog channel is not a perfect simulation of the physical world's effects, it captures many of the effects that appear to affect face detection and has allowed us to get useful results. Our evaluation of the analog channel is described later (§2.9).

2.7 Oracle

We use the analog channel to build an *oracle* (figure 2.15) that predicts whether a face will be detected after an image is degraded by the physical channel.

Naively, we might simply apply the analog channel to the image and then apply the Viola-Jones to the result. However, because the noise is partially random, this is not a good

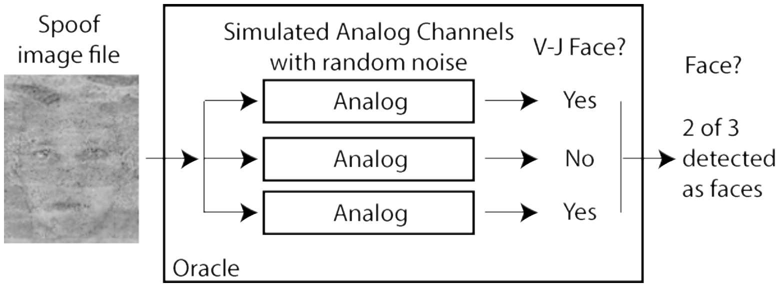


Figure 2.16: Our oracle measures how often a face is detected when repeatedly subjecting the image to the analog channel.

predictor of whether an image will be *reliably* detected as a face: even though Viola-Jones detects a face in the degraded image, the noise might have just been in our favor that one time.

It is more useful to repeat the procedure multiple times. Our oracle passes the image in parallel through several copies of the analog channel, runs Viola-Jones on each result, and reports how many times Viola-Jones detected a face (see figure 2.16).

2.8 Attack: Random Shift (Analog)

We use our simulation of the physical world to improve the attack and generate spoof images that are more robust to degradation and noise from the real world. We use the same simple search procedure as before: pick a random pixel, move its intensity closer to the corresponding pixel in the cover image and discard the change if our oracle reports failure. But now we use our upgraded oracle that uses several copies of the analog channel. Also, for each search, we allow shifting the pixel value, not just 50% toward the cover image, but some shift rate, $s \in (0, 1)$.

Initially, we tested the attack with a naive oracle that only applies the analog channel once. However, we found that the search quickly gets driven towards local minima: it tries a change that actually causes the image to be detected only occasionally (say, 10% of the time), but due to bad luck, the image is accepted by the oracle. Because the search algorithm tries many candidate changes, many of which are bad, eventually it will get unlucky and accept a bad change. Once it has moved towards a bad image, the algorithm is unable to recover. As the search progresses, bad choices vastly outnumber good choices — most choices are bad — so the algorithm has a high probability of going awry.

We next tested an oracle that tries multiple times and requires a face be detected at least 2 out of 3 times (or 4 out of 5 times), but we found this suffers from the same problem. Therefore, we settled on an oracle that applies the analog channel 10 times and requires that a face be detected in all 10 out of 10 trials. We found that this was sufficient to fix the problem.

Algorithm 2: Analog attack

```
Function Search($F$, $C$, $s$, $n$, $m$):
    Input  : Face image $F$,
             Cover image $C$,
             Shift rate $s$,
             Required detections by Oracle $n$,
             Number of tests by Oracle $m$
    Output : Spoof image $S$
    $S \leftarrow F$;
    while not Stalled($S$) do
        $p' \leftarrow$ a random pixel in $S$
        $T(p) \leftarrow \begin{cases} \text{Round}\left(S(p) \times (1-s) + C(p) \times s\right) & \text{if } p = p' \\ S(p) & \text{o.w.} \end{cases}$
        if Oracle($T$, $m$)$\geq n$ then
            $S \leftarrow T$
    return S
Function Stalled($S$):
    Input  : Image $S$
    Output : Boolean
    Return True if $S$ has not changed in last several calls.
Function Oracle($T$, $m$):
    Input  : Test Image $T$
    Output : Boolean
    Invoke Viola-Jones(Analog($T$)) $m$ times and return how many times it detects a
      face with bounds within 10% of those of the face in $F$.
```

Experiment We prepare the input face the same way as in §2.5. We use cover images of granite and all-white and a shift rate of $s = 0.7$. Low shift rates (e.g., $s = 0.05$) take a long time and tend to create a faint but visible shadow of a face. Moderately high shift rates (e.g., $s = 0.7$) and an all-white cover image tend to create abstract dot art. Very high shift rates (e.g., $s = 0.9$) tend to fail quickly. Each run takes a few hours. We have not yet parallelized the analog channel nor used a GPU. The analog channel is expensive, though about a third of the cost is Viola-Jones anyway.

Results Figure 2.17 shows a spoof image generated using this algorithm and an all-white cover image: see the two images in the upper-right. These images do not stand out to us as faces — they look rather like some kind of abstract art — but Viola-Jones detects a face in them, even after applying the physical channel. The image on the upper-right is detected as a face 70% of the time (after the physical channel); the image to its left is detected as a face

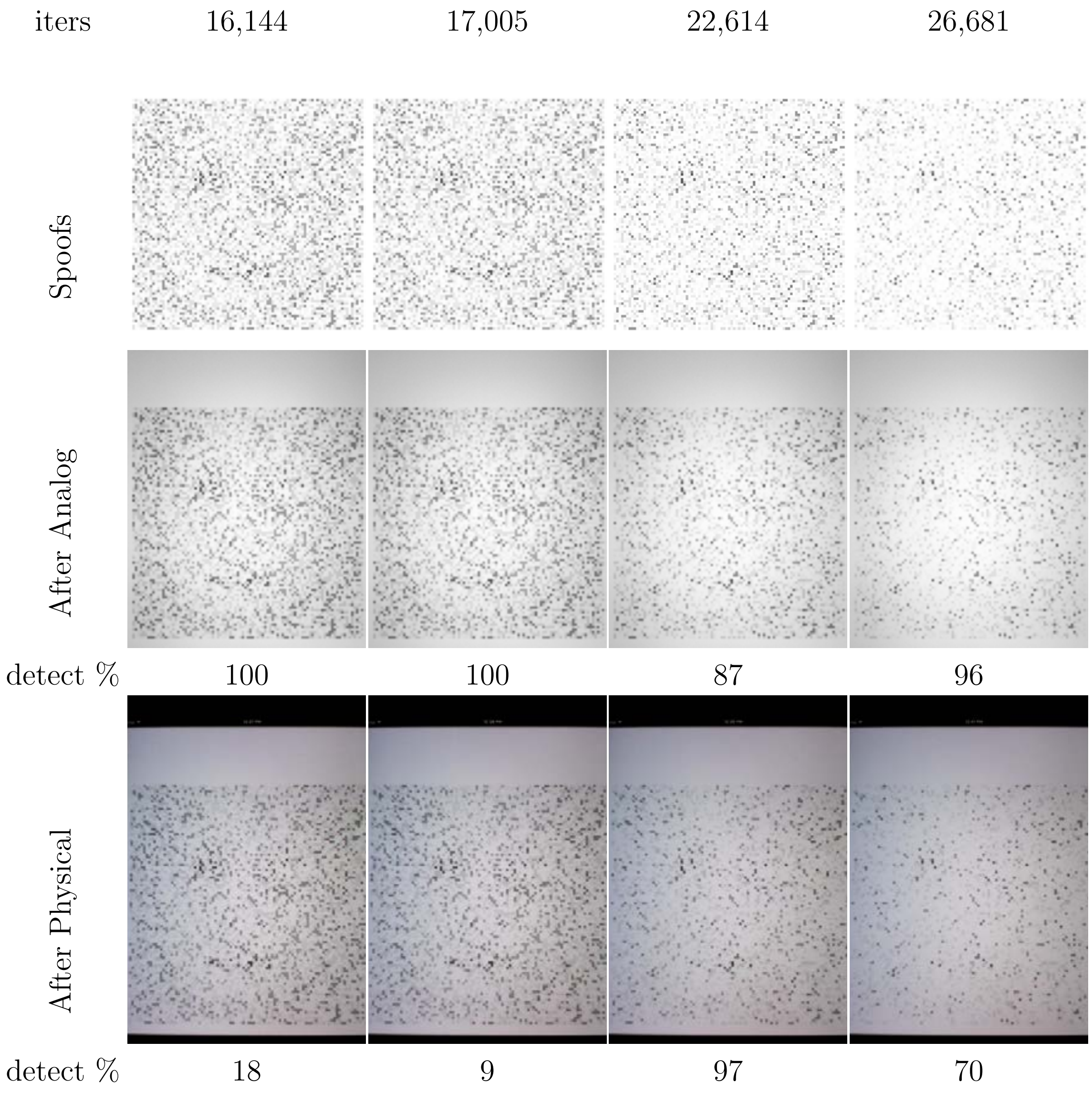


Figure 2.17: Spoof images generated by our algorithm using an all-white cover image and shift rate $s = 0.7$. We also show an example of applying the analog and physical channel to each image and the detection rates after each channel.

97% of the time, i.e., very reliably.

The four columns in figure 2.17 correspond to different points in time along the evolution of a single run of the search algorithm. We show the number of iterations so far, the current spoof image (i.e., S), an example of applying our analog channel to that image and the detection rate after applying the analog channel many times, as well as an example image after applying the physical channel and the detection rate after applying the physical channel many times.

Running our algorithm with a granite cover image was not successful. It takes 9,000 iterations until the algorithm generates a candidate spoof image that is no longer human-recognizable, but the images stop being recognized by Viola-Jones as faces well before that — after 6,000 iterations, the detection rate after the physical channel drops to zero.

Discussion Our approach is successful at creating images that are often detected by Viola-Jones as faces, but which are not as noticed by humans as faces. The images are not as stealthy to humans as before, but they are more robust: They are detected even after being displayed on a tablet and then captured by a camera.

Our attack uses Viola-Jones solely as a black box, obtaining only a boolean result from it. One can view the randomized analog channel and 10-out-of-10 oracle as a way of obtaining a probabilistic measure of success (a continuous confidence metric) from this black box.

2.9 Evaluation of Analog Channel

To evaluate the effectiveness of our simulated analog channel, we ran many images through both the analog channel and physical channel to compare the face detection rate after each. We gathered 1200 images seen during runs of our algorithm. For each image, we fed it through the physical channel 100 times and counted how many times Viola-Jones detected a face in the result. We also did the same for the analog channel. Figure 2.18 shows a scatterplot of the resulting scores. We see that the score from the analog channel is an imperfect but useful predictor of the physical channel: the analog channel helps us rule out some images that won't survive the physical channel, but is sometimes too optimistic about the likelihood that Viola-Jones will detect a face after the physical channel is applied. This explains why our oracle helps improve the search algorithm: while not perfect, it provides feedback to help the random search avoid some images that certainly won't survive the physical channel.

2.10 Failed Attack: Gradient Descent

Before arriving at the simple algorithm described earlier, we tried other approaches. Most notably, we tried an approach inspired by gradient descent, where we tried to measure which pixels Viola-Jones is most sensitive to.

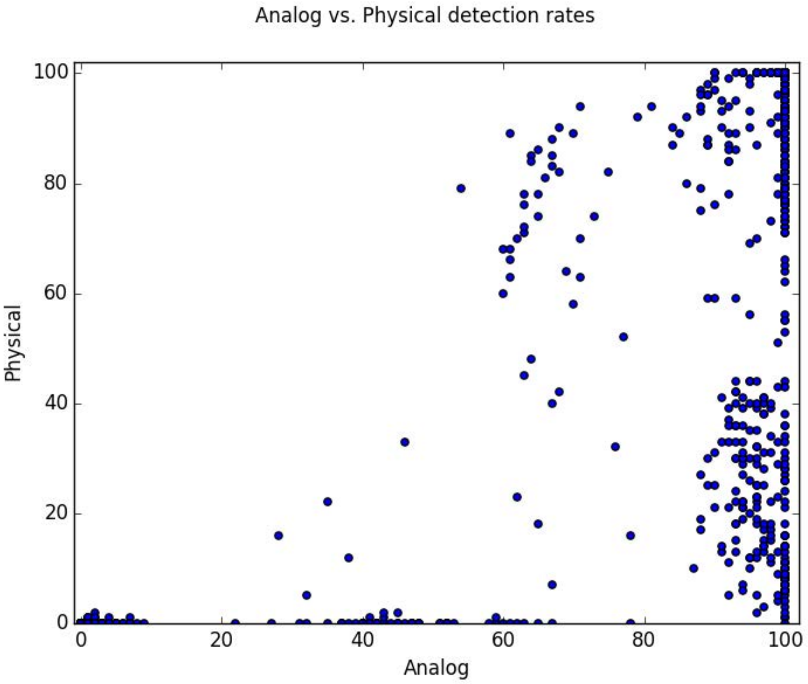


Figure 2.18: Evaluating how well the analog channel models the physical channel. We plot the face detection rate after the physical channel vs. face detection rate after the analog channel, for 1200 different images seen at different iterations of our algorithm. The analog channel helps avoid some images that would fail the physical channel, but far from guarantees success: the physical detection rate is generally at most the analog detection rate, but often substantially smaller.

To approximate the gradient of the detector's confidence that the image is a face, we first found a blend of the current image and the cover image that was right on the edge of Viola-Jones' decision boundary. In particular, we used binary search to find the largest real number $\alpha \in [0, 1]$ such that the blend $(1 - \alpha) \times S + \alpha \times C$ was still detected by Viola-Jones, then reduced α slightly and set $S' = (1 - \alpha) \times S + \alpha \times C$. This gave us an image S' in the search space close to the decision boundary. We then picked small regions of pixels and saw how close to the cover image we could move them as a group before Viola-Jones failed. Doing this for overlapping regions gave us a measure of how sensitive Viola-Jones was to changes to each pixel. In this way, we built an approximate sensitivity map M, where $M(p)$ is proportional to how far we can move pixel p of T towards $C(p)$ before Viola-Jones stops detecting a face (i.e., larger values of $M(p)$ indicate lower sensitivity to changes to p). Then, we used this information to move the original image S closer to the cover image, weighted to move the less sensitive pixels the most: i.e., we replaced S with the image T defined by

$$T(p) = (1 - \varepsilon \cdot M(p)) \times S(p) + \varepsilon \cdot M(p) \times C(p), \tag{2.7}$$

for some small constant $\varepsilon > 0$. We iterated this process until convergence.

While slow, this process created images with just the key face features left. Unfortunately, those images were readily human-recognizable as faces; see, e.g., figure 2.19. The randomness of our current approach seems to hide the face better.

Figure 2.19: An image generated using our gradient descent-inspired attack.

2.11 Related Work

There has not been published work on stealthy spoofs of 2D face detection. Work in the opposite direction has tried to hide from facial detection, from obvious makeup, styling, or partial obscurement to thwart detection[Har10]; selective feature identification and face art to thwart detection and recognition[FP13]; and subtle makeup to thwart recognition[EKD13].

Recent work on physical world attacks on object recognition measured the degradation in the effectiveness of adversarial images when they are first printed and photographed with a cell phone[KGB17]. Though they measure the effects of components of that physical channel, they construct their images using knowledge of the object detection algorithm, and not the channel. Our work constructs adversarial images without knowledge of the detection algorithm.

In the related domain of voice, spoofing attacks have been demonstrated that create sound that is unintelligible as voice by humans yet parsed as voice commands by voice recognition software[CMV$^+$16].

Much work has been done on using liveness detection as a defense mechanism, including work based on spectral analysis[LWTJ04] and the differences between how a live face shifts and the movement of a static printed image[MHP11]. Yet, 2D video created from a virtual reality model has been used to defeat several current and proposed liveness detection systems[XPFM16]. Though we have not addressed liveness detection, defeat of Viola-Jones as the first step of the defense chain did not require sophisticated techniques; incorporation of video-based liveness spoofing would be interesting future work.

2.12 Conclusion

We have shown that deliberate spoof images can be created that do not appear to humans as faces, yet Viola-Jones often detects as faces, even after passing through a simulated physical world. This indicates that facial detection can be fooled, and in a way that human observers are unlikely to notice as suspicious.

Chapter 3

Background Class Defense

3.1 Introduction

Deep neural networks have been very successful at many classification tasks. Yet they have been found to be vulnerable to misclassifying deliberately crafted images [SZS$^+$14, GSS15].

We examine whether adding a large and diverse background class to the training data can help detect and neutralize adversarial examples.[1] We assume that there is a set of key classes that we care about distinguishing between. We propose to train the classifier by adding additional training images that are not from the key classes that we care about distinguishing between. We introduce a new class that we call the background class for these additional images. The background class effectively serves as a "none of the above." For example, if we want to recognize MNIST digits, we might use images of non-digit handwriting as examples of the background class. The background class is intended to create a default classification to provide better separation between the key classes within the model's feature space. We measure how this defense affects the classification of normal images and the detection of adversarial examples.

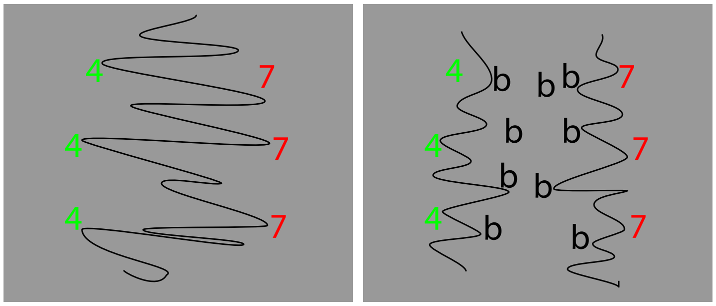
Figure 3.1 illustrates a simplified visualization of the decision boundary of an undefended classifier (left) and our proposed defense (right). We hope that the background class makes it harder to modify an image from one key class to be misclassified as another key class.

These background images are not noise, but images of objects distinct enough from our key classes to be distinguished from them by the model.

In use, our network is still evaluated on its ability to distinguish between images that humans perceive as key classes. We change the classification task from image → one of the key classes, to image → one of the key classes or background.

We evaluate this idea with the EMNIST dataset [CATv17], of handwritten digits and letters, and the fast gradient sign [GSS15], fast gradient [LCLS17], and Carlini [CW17a] attacks.

[1] "Background Class Defense Against Adversarial Examples", M. McCoyd and D. Wagner, IEEE DLS 2018



(a) The original classifier (b) With background class (our defense)

Figure 3.1: The intuition behind our defense. In (b), we introduce an extra class, the background class, for images other than the classes we want to recognize. The hope is that this separates those classes enough that adversarial examples are no longer effective.

We find our defense aids detection of simple attacks, but we have not found any increase in detection of state-of-the-art attacks. There are weaknesses in our examination; in particular, the letters of our background class are not crafted for the role of separating the digits in image or feature space.

3.2 Background

Neural networks have been very successful at learning a function, $F_\Theta(x)$, by tuning a set of weights, Θ, based on training data, x, with labels, y, that allow them to then accurately label new data. For m-class networks, y is a probability distribution over m labels. These networks are trained by defining a loss function, $R(\Theta)$, that defines how far off their predicted labels for the training data are from the training data's actual labels. The loss function is evaluated on the training data. Backpropagation [RHW88] is used to allocate blame for the error in each of the network's outputs. This output error is propagated backward through the network based on the current weights on each neuron's inputs. This allows the weights at each layer to be adjusted appropriately in the direction of reducing the cost, $\partial_\Theta R_\Theta(x)$.

Deep Neural networks have improved neural network performance, achieving near human-level performance on several visual tasks, including visual recognition of objects [RDS+15] and road signs [SSSI11], leading to their increased use.

Adversarial Examples

It has been found that an attacker with knowledge of the model can construct, from any normal image, x, an adversarial example [SZS+14, GSS15], $x^\star$, that looks to humans like the normal image's classification but that the model classifies differently from the normal image.

In an *untargeted* attack, the attacker might strive for these images to merely be classified by the model differently from how humans would classify them. In a *targeted* attack, they might strive for them to be classified as a specific class. A recent overview of adversarial examples can be found in [CW17a].

Two important aspects of adversarial examples are that 1) they are classified by humans and the model differently, and 2) we care about that misclassification for some task. Our defense will exploit the second of these.

Three quantitative metrics are commonly used to measure how much an attack image differs from the original image, the L_0, L_2, and L_∞ distance metrics. L_0 is the number of pixels changed. L_2 is the square root of the sum of the squares of the changes to each pixel. L_∞ is the maximum change made to any pixel.

Attacks

The basic approach of finding adversarial examples can be seen as a variation on the normal backpropagation training of the model weights. Instead of propagating the error back through the network to ascribe proportional blame to the weights, we ascribe proportional blame to each input pixel. For each pixel, we obtain a contributory sign and magnitude for the errors in the output neurons. Attack methods vary in how they use this information. All attack images must be clipped as needed to remain within the range of valid pixel values. There are several attacks; we highlight only a few.

Fast Gradient Sign One of the simplest attacks to understand is the fast gradient sign attack [GSS15]. Using backpropagation, it finds the gradient of the loss function with respect to the inputs, simplifies that to just the sign of the gradient[2], and then moves by a factor τ in that direction:

$$x^\star \leftarrow x + \tau \, \mathbf{sgn}(\nabla_x R(F(x))).$$

Fast Gradient The fast gradient attack [LCLS17] normalizes the gradient to have an L_2 norm of one and then moves by a factor τ in that direction:

$$x^\star \leftarrow x + \tau \, \frac{\nabla_x R(F(x))}{\|\nabla_x R(F(x))\|_2}.$$

Carlini The Carlini attack [CW17a] is a strong iterative attack that reduces the set of pixels it uses by repeatedly eliminating the least effective pixels from the set it uses. It produces attack images with very small changes, based on either L_0, L_2, or L_∞ distances.

[2] Normalizing to an L_∞ norm of one.

(a) Original image, classified as a 9.

(b) Attack image, classified as a 7.

Figure 3.2: Fast gradient untargeted L_2 attack with $\tau = 6.79$.

(a) Original image, classified as a 2.

(b) Attack image, classified as a 3.

Figure 3.3: Carlini targeted L_2 attack, adapted from [CW17a].

Defenses

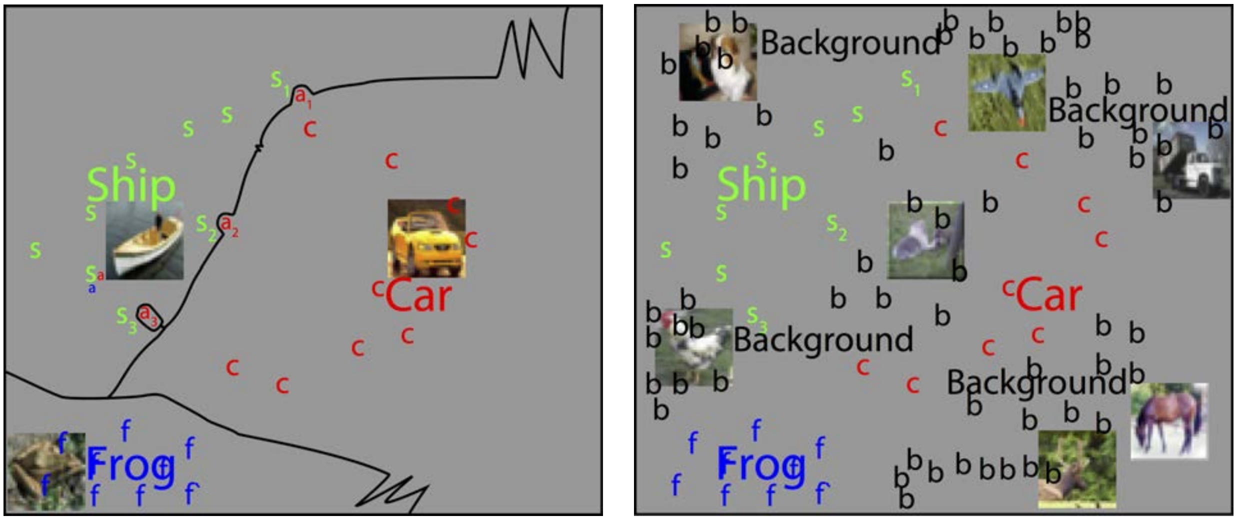
Several defenses have been proposed, but none so far are effective. One of the most natural defenses is adversarial retraining. [GSS15] found that an MNIST network trained with fast gradient sign adversarial examples reduced the success of that attack from 89.4% to 17.9%. [SYN15] retrain based on a gradient step of L_1, L_2, or L_∞, and see increased robustness to fast gradient sign attacks, while achieving increased accuracy on normal images due to regularization, on MNIST and CIFAR-10 datasets. Yet unpublished work by others find that such defenses are vulnerable to attack. They find that for MNIST, retraining with images generated with the fast gradient sign attack only increases the mean minimal distance of adversarial examples by a factor of two measured using the Carlini L_2 attack.

3.3 Background Class Defense

Without a diverse background class, there are large areas of the feature space that are not represented in the training data. In these unrepresented parts of the feature space, there is little to drive the model toward a smooth decision boundary. Our rationale is that having no classifications besides the ones we care about may also contribute to not having the smoothness we would prefer in the areas around our key classes.

For our work, there are two key aspects to adversarial examples: 1) for humans, they are perceptually close to other images that the model classifies differently, and 2) we care about that misclassification. We hope to manipulate the model so that when perceptually close images are classified differently, we no longer care as much, as the classification difference is between a key class and the background class instead of between two key classes. Such misclassification as the background class alerts us that we are under attack.

Illustrating Example As an example, we might want to distinguish between ships, cars, and frogs, as in figure 3.4. With normal training, the model learns to separate the key classes, yet the decision boundary may have a very erratic shape in areas with no training data, and the boundary includes adversarial examples around all the training data points.



(a) With three labels: Ship, Car, Frog

(b) With four labels: Ship, Car, Frog, Background

Figure 3.4: Example classification task. (a) with a potentially erratic decision boundary in areas with no data and adversarial examples including a_1, a_2, a_3, (b) with the addition of background class training data.

Our training adds one extra label 'background' for a diverse set of background images added to the training data, e.g., birds, trucks, planes, cats, dogs, horses, and deer, as in figure 3.4b. We hope that the model will find it cheapest to learn a decision boundary that connects all these background images as one area, restricting the key class areas to be around their training data. The goal is that around images perceived as key classes by humans, such as S_2, the decision boundary might still not be smooth, but there would no longer be adversarial examples misclassified as other key classes, but only images misclassified as background, a misclassification we are less concerned about. Such background classifications might serve as a flag of some failure in our input or of an attack.

3.4 Data

We base our evaluation on the EMNIST dataset, chosen for ease of training. EMNIST [CATv17] expands MNIST to include letters. It contains 28×28 grayscale images of handwritten digits and upper and lower case letters, in 62 classes, with 697,932 training images and 116,323 test images. In our EMNIST tests, the digits serve as our key classes and the letters collectively as the background class. We adjust pixels to the range [-0.5, 0.5].

EMINST has different versions depending on whether you want digits and/or letters and whether you want the classes to be balanced. The Balanced set has digits and letters with balanced class sizes. The By Merge set is roughly balanced within the digits but very unbalanced within the letters; it is, however, larger. Both of these datasets have 47 classes as they merge several letters whose lower/upper appearances are similar, the 15 letters 'c i j k l m o p s u v w x y z'.

Confusing Letters

As a source of background images, letters have a problem: some look to humans like digits. We are trying to use the background class to fill in the part of the input/feature space that is away from the key classes and thus normally unrepresented. Yet pairs such as 0/O and 1/l look alike. Using such letters in our background class brings the edges of the background class closer to our key classes than we may need for our defense.

As a data preparation step, we pick letters for our background class that are very distinct from digits. We measure digit/letter confusion by creating a confusion matrix using the original digit and letter labels. For each letter, we measure how often it is classified as any letter or 'misclassified' as any digit. We keep for our background class the 22 letters, 'T t U d E P f F X H C n h A K M V W e R r N', that are classified as letters $98 - 100\%$ of the time, an arbitrary cutoff. We remove from the dataset the remaining 15 letters, 'O L q I g Z S Y G b D B J a Q', that are classified as letters only $69 - 97\%$ of the time. To measure this, we use 10-fold cross-validation repeated four times on the Balanced set trained for 20 epochs. We used the Balanced set for these tests because the larger By Merge set caused memory issues.

Digits + Background Dataset

We train our defense with the By Merge dataset, the largest EMNIST dataset, stripped of the confusing letters found above. All letters are merged into a single background class, so the classifier classifies every image into one of 11 classes (0-9, or background). This gives us a dataset for the train/validation splits with class counts of between 31,280 and 38,304 for 0-9 and a class count of 212,657 for the background class, with a count of 2,535 for the least frequent letter and 24,657 for the most, summarized in table 3.1.

We found that using more of the confusing letters during training increases detection of attack, yet decreases accuracy when not under attack. In the extreme, it must learn that zero and the letter 'O' belong to different classes. We do not have enough results on that tradeoff to report more fully.

Table 3.1: Data sizes

Baseline classifier (no background class):	
Train:	310,912 digits
Test:	34,432 digits
With background class defense:	
Train:	310,912 digits + 191,360 letters
Test:	34,432 digits

3.5 Standard Model and Training

Our EMNIST model is very simple. It is the example MNIST model of the Keras framework with dropout replaced by batch normalization, described in table 3.2. A ReLU activation function was used for the convolutional and first fully connected layers. Logits are produced by the final layer. We did not do any tuning of model parameters. We simply picked common values for kernel, stride, and layer outputs.

Table 3.2: EMNIST model's architecture

Layer	Kernel	Outputs	Activation
Conv	3×3	32	BatchNorm + ReLU
Conv	3×3	64	BatchNorm + ReLU
MaxOut	2×2	64	
Dense		128	BatchNorm + ReLU
Dense		11	BatchNorm

Training was done with the Ada Delta optimizer[Zei12] with a cross-entropy loss function and a batch size of 128.

Results Before Attack To measure the cost of the defense, we examine the model's classification rates for normal images when trained on the data sets, before examining changes to the robustness to adversarial examples. Training and testing were done with 10 random 90:10 splits of the train dataset for 10 epochs each.

The accuracy is shown in table 3.3. We obtain 99.6% accuracy when trained on digits, and 99.1% when trained on digits and letters. These results are not state of the art for MNIST, but those trained with the background class are still near the range of performance for which we turn to deep models.

Table 3.3: Classification accuracy for normal images (not under attack)

Dataset	Mean Accuracy (%)		
	Correct	Other	Background
Digits	99.62	0.38	
Digits+Background	99.11	0.41	0.48
Change	-0.51	+0.03	+0.48

3.6 Attack Results

We use three attacks from the literature in our evaluation, two weak attacks and one strong. Fast gradient sign has been used in the past to evaluate defenses, though it is now considered a weak attack. Fast gradient is an L_2 version of fast gradient sign. The Carlini attack is a strong L_2 attack that has defeated several proposed defenses in the literature.

Table 3.4 summarizes the difference that the addition of our background class has on these attacks. In each case, the attack has full access to the trained model under attack and uses images from the EMNIST dataset. We do not see an increase in security against the Carlini attack. We discuss these attacks below.

Table 3.4: Attack Success Rates

	Attack		
Dataset	FGS	FG	Carlini
No background	92%	94%	100%
Background	68%	56%	100%
Change:	-24%	-38%	0%

Weak Attacks

As an initial test of our defense, we use fast gradient sign and fast gradient, with epsilons of 0.25 and 6.79, respectively. Fast gradient with $\epsilon = 6.79$ produces an L_2 difference similar to fast gradient sign with $\epsilon = 0.25$. As before, training and attack were done with 10 random 90:10 splits of the train dataset for 10 epochs each.

Table 3.5 reports details of whether each attack image is classified with its correct original classification, some other key classification, or the background classification. Fast gradient

Table 3.5: Classification rates for adversarial images (weak attacks)

Attack and Dataset	Mean Classification %			Standard Deviation		
	Correct	Other	Background	Correct	Other	Background
Fast Gradient Sign						
Digits	8.26	91.74		1.92	1.92	
Digits+Background	2.26	68.39	29.35	0.99	12.50	13.22
Change	-6.00	-23.35	+ 29.35			
Fast Gradient						
Digits	5.91	94.09		1.42	1.42	
Digits+Background	1.21	55.99	42.80	0.80	6.82	7.11
Change	-4.70	-38.10	+42.80			

sign shows a reduction in attack success from 92% to 68% when the background class is used; the attack is detected 29% of the time when we use the background class. Fast gradient shows a reduction in attack success from 94% to 56% when the background class is used; the attack is detected 43% of the time when we use the background class. These are significant reductions in attack success.

Strong Attack

One of the strongest current attacks is the targeted Carlini attack. We evaluate our defense against it, training and testing on a smaller version of the EMNIST dataset, the balanced dataset instead of the by merge dataset.

Our results show no increase in the detection of the Carlini attack. The Carlini L_2 attack succeeds 100% of the time for both a dataset of digits and a dataset of digits plus background.

Because we have not found an increase in the detection of strong attacks, we have not used the testing portion of the EMNIST dataset to produce our results.

3.7 Limitations

We have not found the approach to work, but have not pushed hard at it. These are some possible changes.

Strain the model's capacity

In adding a large background class, when does the model architecture have to change to preserve accuracy on key classes? We did not change the model architecture for our work, other than switching it to use batch normalization in an effort to avoid having to tune hyperparameters to maintain normal image accuracy. Yet, a key hypothesis in our work is that adding the background class will force the model to learn a default classification of background for the area between the key classes. We hoped this would be the cheapest thing to learn and that the model could not afford to learn anything more complex. Yet we have not pushed at keeping the capacity of the model under pressure. How do the results change if we remove capacity from the network until the point when we can no longer maintain normal image accuracy?

Background images close to and between key classes

We used letters as background images for simplicity. Yet, with random locations, there is little guarantee that they fill the crucial space between our key classes. If the idea behind our approach is a good one, ideally, we would like to sprinkle background images throughout the non-key parts of the space. And especially all along the border of each key class, but a bit separated from it so as to not degrade performance on images within the key class, such as figure 3.1.

One approximation of the around/between and close part would be to use various morphing algorithms to change images from one key class into images of another key class. Use a human to judge when the image stops being the first class and when it starts being the second class. Test that this method does not cause your image paths to cross over any other key classes along the way. Hopefully, you find a pattern for these change points, such as 30% and 70% along the way, and use those points, or just 50%, as a new background image. As you fill in the space, classification of new ones as background images might reliably allow you to avoid creating auto-generated background images that were actually key class images. A complication is that each key class may have several separate areas in the feature space.

A possible experiment to test this would be to pick a small subset, or one, of our key classes and use this to sprinkle the area surrounding those classes with close background images. Then evaluate whether attacks involving this subset were more difficult than attacks involving only other key classes.

Such artificial construction of background images is almost certainly required for image spaces with higher dimensionality or with far more classes, though at some point, this may cause the background images to no longer appear normal, being just a synthetic morph between two points in the image space.

Use more background images

We could expand the types of images used for the background class or use standard data augmentation techniques. For our background images, we stayed within the same domain that our model had to learn, simple things that were handwritten. This was deliberate in an attempt to not change too much what the network needed to learn but instead to just add a new large class. Staying within that domain, letters could be chopped into subareas and rearranged to form random 'characters' that were not digits. This would add extra sharp edges that are not otherwise present in our data. Using random images – trees, cars, faces – would expand the data set even more. Also, the more complex domains, represented by such datasets as CIFAR, GTSRB, and ImageNet, likely do not have a simple and similar domain from which to draw such images, so use of more random background images would be necessary for our method in harder image domains.

3.8 Related Work

Bendale and Boult [BB16] examine detecting images from the part of the image space not trained on, which they call the open set. They examine open set and fooling images, detecting and rejecting images that look to humans like an unknown class. They do not examine adversarial images. Their open set is essentially the same as our background class. However, a crucial difference is that they do not use open set images in training, while we train with images from the background class to try to avoid adversarial examples.

Melis et al. propose a similar defense, where the classifier has the option to reject an image as not belonging to any of the trained classes [MDB+17]. During training, the classifier learns a region for each class that encompasses most or all of the examples of that class in the training set. At test time, the classifier rejects the image if it does not fall into any of these regions. Thus, their defense can be considered another open-set scheme. Like Bendale and Boult, they train only on normal images, but do not use open set images in training; in contrast, our defense trains on examples of the background class.

Goodfellow et al. discuss rubbish classes, "degenerate inputs that a human would classify as not belonging to any of the categories in the training set", and the problem of not classifying such images as belonging to one of the expected classes [GSS15]. They discuss rubbish classes in connection with the fooling images of Nguyen et al. [NYC15], abstract or apparently random images that are classified with high confidence. Goodfellow et al. show that on CIFAR-10 it is easy to create a fooling image from a random image by adding a few gradient steps toward a target class. Neither group trains the classifier on such rubbish images, nor do they discuss normal images that are just not in the classes of interest. We train on background class images that are normal, not random, images.

Hosseini et al. introduce NULL labeling to mark adversarial examples added in training, so that under attack labeling shifts to the NULL class [HCK+17]. We hope for a similar shift, but without relying on distinguishing adversarial examples. We rely on background being the available misclassification.

3.9 Conclusion

Adding a background class to image classification training significantly increases the detection of simple adversarial example attacks, but we have not found it to stop advanced attacks. For the weak attacks, we have reduced the success rate significantly, suggesting that we have filled the space between our key classes with the background class. For the strong Carlini attack, we see no effect, suggesting that the trained model is still very convoluted near each of the training examples, and still vulnerable to attack. While not a successful defense in itself, adding the background class to the training data may be a useful tool. Further work could investigate constructing background images between the key classes and artificially expanding the background data.

Chapter 4

Minority Reports Defense

4.1 Introduction

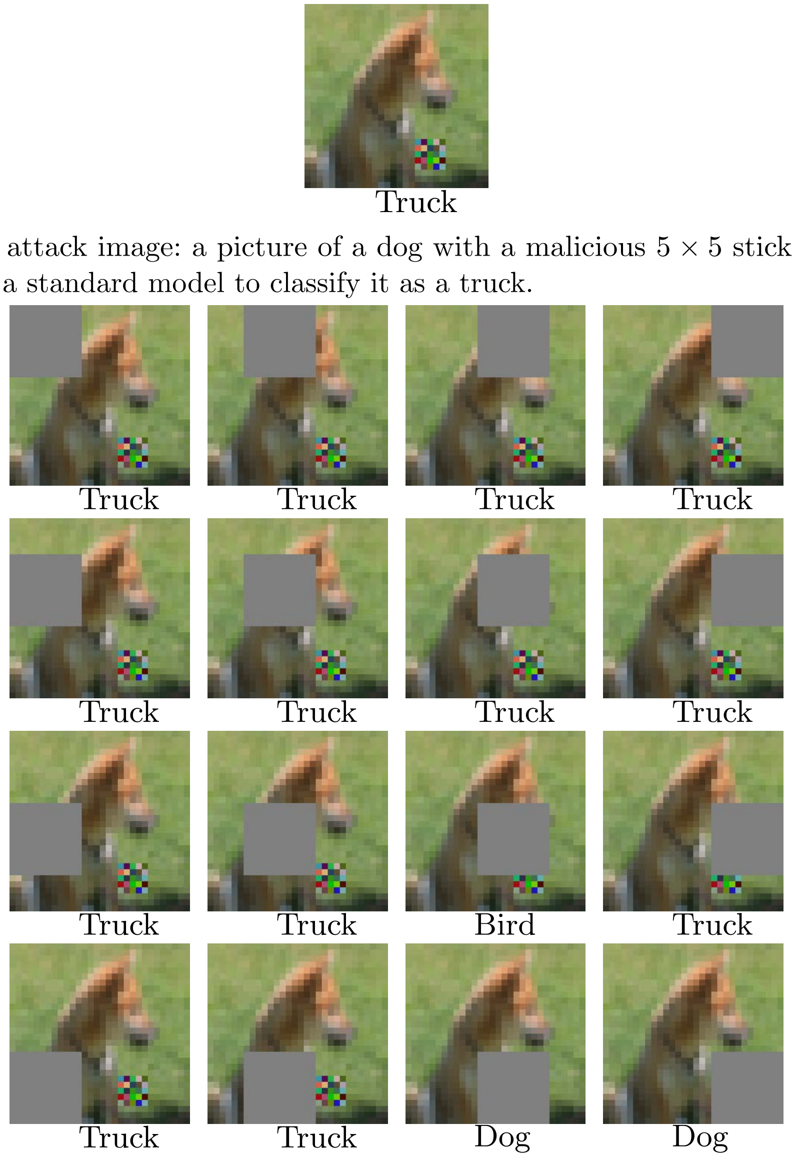
Deep learning image classification is widely used yet is vulnerable to adversarial attack, which can change the computer classification without changing how humans classify the image. An attacker with knowledge of a neural network model can construct, from any normal image x, an adversarial example $x^{\star}$ that looks to humans like x but that the model classifies differently from the normal image [SZS+14], [GSS15], [HJN+11], [CW17b].

Recently, researchers have proposed the adversarial patch attack [BMR+17], [KZG18], where the attacker changes just a limited rectangular region of the image, for example, by placing a sticker over a road sign or other object. Others have expanded on the vulnerability to this type of attack [EEF+17], [TRG19], [XZL+19]. In this paper, we propose a defense against this attack.

The idea of our defense is to occlude part of the image and then classify the occluded image. [1] First, we train a classifier that properly classifies occluded images. Then, if we knew the location of the adversarial patch, we could occlude that region of the image (e.g., overwriting it with a uniform grey rectangle) and apply the classifier to the occluded image. This would defend against patch attacks, as the attacker's contribution is completely overwritten and the input to the classifier (the occluded image) cannot be affected by the attacker in any way.

In practice, we do not know the location of the adversarial patch, so a more sophisticated defense is needed. Our approach works by occluding an area larger than the maximum patch size and striding the occlude area across the image, making an occluded prediction at each stride. We then analyze the classifier's predictions on these occluded images. If the occlusion region is sufficiently larger than the adversarial patch, several of the occluded images will completely obscure the adversarial patch and thus the classifier's prediction on those images will be unaffected by the adversary and should match the correct label. Thus, we expect the correct label to appear multiple times among the predictions from occluded images. We

[1] Minority Reports Defense: Defending Against Adversarial Patches. Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Liu, and David Wagner. SiMLA 2020.



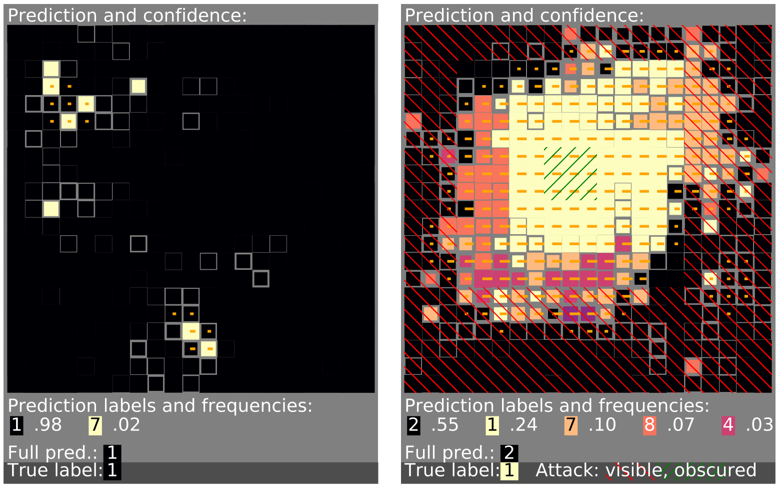
(a) An attack image: a picture of a dog with a malicious 5 × 5 sticker that causes a standard model to classify it as a truck.

(b) We occlude part of the image with a grey square, then classify these occluded images. Here the 3rd and 4th predictions in the 4th row will be unaffected by this attack. Our actual defense ensures that any attack will be fully occluded by a 3×3 grid of predictions, instead of the 1×2 grid shown here.

Figure 4.1: Our scheme works by occluding different portions of the image and analyzing the predictions made by the classifier on these occluded images.

show how to use this redundancy to detect adversarial patch attacks. We call our scheme the minority reports defense because no matter where the patch is located, there will always be a minority of predictions that cannot be influenced by the attacker and vote for the correct label.

Figure 4.1 illustrates our defense. We take the input image (Figure 4.1a) and construct a grid of partially occluded images (Figure 4.1b) with occlusions at different locations, chosen so that any attack will be occluded in a cluster of several predictions. We then apply the classifier to each occluded image to obtain a grid of predictions. When under attack, we can expect most predictions to differ from the true label, but there will always be a cluster of locations where the adversarial patch is fully obscured, and thus the labels are all expected to agree with the true label; in Figure 4.1, the 3rd and 4th images in the 4th row obscure the adversarial patch and thus vote for the true label. Our defense analyzes the grid of predicted labels to detect this pattern. If there is a cluster of predictions that all match each other but are in the minority for the prediction grid overall, then this suggests an attack. Figure 4.2 visualizes the prediction grid for a benign image (on the left) and a malicious image containing an undefended adversarial patch (on the right).



(a) Benign (b) Attack

Figure 4.2: Prediction grids for a benign image (left) and an undefended attack image (right). Each cell in the grid is colored based on the classifier's prediction when fed an image obscured at that position in the grid. A cluster of identical minority predictions, as seen in the right image, suggests an attack. In the attack image on the right, green hashes mark the nine predictions where the adversarial patch was fully occluded.

We evaluate our scheme on the CIFAR-10 [KH09], Fashion MNIST [XRV17] and MNIST [LBBH98] datasets with a stride of one. We show that our defense does not harm accuracy much. We also evaluate its security against adaptive attacks. In particular, we show how to bound the success of any possible attack on a given image, and using this, we are able to demonstrate certified security for a large fraction of images. In particular, we are able to prove a security theorem: for a large fraction of images in the validation set, we can prove that no patch attack will succeed, no matter where the patch is placed or how the patch is modified, so long as the size of the patch is limited. In summary, our contributions in this paper are:

- We quantify the vulnerability of undefended networks for Fashion MNIST and MNIST against patch attacks with patches of different sizes (§4.2).
- We propose a novel method for detecting patch attacks, based on differently occluded views of the input image (§4.3).
- We provide a worst-case analysis of security against adaptive attacks for CIFAR-10, Fashion MNIST, and MNIST (§4.4 and §4.6).

4.2 Patch Attack

Patch attacks [BMR+17] work by replacing a small part of the image with something of the attacker's choosing, e.g., by placing a small sticker on an object or road sign. Figure 4.1a shows a patch attack. Patch attacks represent a practical method of executing an attack in the physical world. Digital images can be manipulated throughout the entire scene they present, yet this is impractical in a physical, not digital, scene. It is far more practical to add an attacker-controlled object to part of the scene.

As a simple, non-malicious example, it is not uncommon to see stickers on road signs in the real world, without preventing humans from understanding the signs or prompting the patch's immediate removal.

Attack Model

We assume the attacker knows everything the defender knows: the architecture, weights, training data, and algorithm of all models and methods used by the defender. The attacker may place a 'compact' patch anywhere within the digital image and arbitrarily modify all pixels within the patch to any values in the pixel range. For simplicity, we restrict the attacker to a patch contained in an $n \times n$ square area for some n, n being a measure of the attacker's lack of stealth, and 'compact' meaning square.

The size of the adversarial patch that can be defended against can be thought of as similar to the size of an adversarial example L2 perturbation that can be defended against. Certainly, the attacker could make a larger change, but at some point, the change either becomes very obvious or changes the meaning of the image for humans. Thus crossing the fussy boundary from being an adversarial, stealthy, attack to being an image of something completely different to humans – thus no longer adversarial as described in §4.1

Patch Sizes

We first study how large of a patch is needed to successfully attack undefended Fashion MNIST and MNIST. We test multiple patch sizes and measure the attacker's success rate for each patch size.

Setup We conduct a targeted attack against standard Fashion MNIST and MNIST models from §4.6. We attack the first 300 validation images for Fashion MNIST and the first 100 validation images for MNIST. We report the fraction of images for which we can successfully mount a patch attack. For each image, we select a target label by choosing randomly among the classes that are least likely, according to the softmax outputs of the classifier (namely, we find the least likely class, identify all classes whose confidence is within 0.1% of the least likely, and select the target class uniformly at random among this set). That target is used for all attacks on that image. For each base image and its chosen target class, we enumerate all possible patch positions and try at each position to find an attack patch at that position.

Attack Algorithm To generate patch attacks, we iterate over all possible locations for the patch and use a projected gradient descent (PGD) attack for each location. We consider the attack a success if we find any location where we can place a patch that changes the model's prediction to the target label. The resulting adversarial patch is specific to one specific image and one specific location.

The standard PGD attack uses a constant step size, but we found it was more effective to use a schedule that varies the step size among iterations. In our experiments, a cyclic learning rate was more effective than a constant step size or an exponential decay rate, so we used it in all experiments. We used a cyclic learning rate with ten steps per cycle, with step sizes from 0.002 to 0.3, for a maximum of 150 steps. We stopped early at the end of a cycle if the attack achieved confidence 0.6 or higher for the target class, or if the confidence had not improved by at least 0.002 in the last 20 steps from the best so far. For each image, we attacked in parallel across all possible patch locations.

Results For our MNIST model, a 6×6 patch is large enough to attack 45% of the images successfully. The success rate for 4×4 patches was 19%, and for 8×8 patches, 80%. When an image can be attacked, there are often many possible locations where an adversarial patch can be placed: for a 6×6 patch, out of all images where a patch attack is possible, there were, on average, 41 different positions where the patch can be placed.

For our Fashion MNIST model, the success rate for patch attacks was as follows: 4×4 patch: 27% success, 5×5 patch: 50% success, 6×6 patch: 60% success.

These results indicate that, on MNIST, an attacker needs to control a 6×6 patch to have close to a 50% chance of success, while a 5×5 patch is large enough for Fashion MNIST, occupying 5% and 3% of the images respectively.

We use 5×5 patches for CIFAR-10, Fashion MNIST and MNIST, as that size is used by recent work [CNA+20].

4.3 Our Defense

The basic idea of the minority reports defense is to occlude part of the image and classify the resulting image. If the occlusion completely covers the adversarial patch, then the attacker will be unable to influence the classifier's prediction. We don't know where the adversarial patch might be located, so we stride the occlusion area across the image. Because we use an occlusion area sufficiently larger than the adversarial patch, no matter where the adversarial patch is placed, there should be a cluster of occlusion positions that all yield the same prediction.

Occlusion Training

As our defense will internally use partial occlusions of the image it is given, we train, or retrain, with occluded images. Each time an image is presented in training, a randomly placed $n \times n$ square is occluded, and the model receives the occluded image. This is similar to cutout training from Devries et al. [DT17], who used occlusion as a regularizer. The difference in our training is that the occlusion is the size we will use in our defense. We also internally provide the model an additional input of a sparsity mask that indicates which pixels are occluded.

For instance, the input to an MNIST model is an image, with dimensions $28 \times 28 \times 1$, and a mask, with dimensions $28 \times 28 \times 1$. The image has its normal channels, and the mask has one channel. In the mask, a 0 indicates an occluded position, and a 1 a non-occluded position.

If a model already predicts accurately with a random partial occlusion of the size we use, there is no need to retrain or modify it, it can just be wrapped in our defense as described in the following sections.

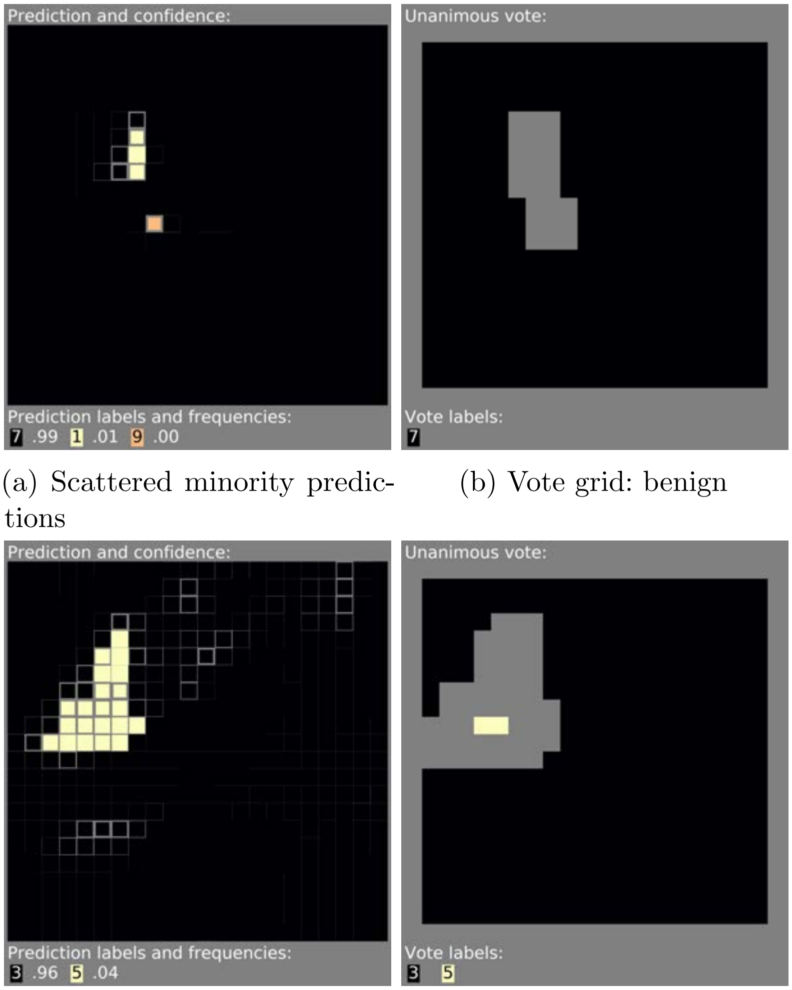
To better handle the missing pixels, we modify the architectures we test by replacing convolutions with sparsity invariant convolutions [USS+17]. If the mask indicates no occlusions, the sparsity invariant convolutions behave as normal convolutions, but when occlusions are indicated, the occluded pixels are handled better.

Training on occluded images appears to have only a small change on the accuracy of the inner model on non-occluded images, see §4.7.

Creating a Prediction Grid

At evaluation time, our defense's first step is to generate a prediction grid as follows. We describe the simpler case of low-resolution images here, leaving the larger stride for higher resolution images to §4.5. For defending MNIST images against a 5×5 adversarial patch, we use a 7×7 occlusion region. We slide the 7×7 occlusion region over the 28×28 image with a stride of one pixel, yielding 26×26 possible locations for the occlusion region. This ensures any patch is covered by nine occlude areas, even a patch at the image edge, $26 = (28-(7-1))+2+2$. The prediction grid is a 26×26 array that records, for each location, the classifier's output. At each location, we mask out the corresponding occlusion region of the image, classify the occluded image, obtain the confidence scores from the classifier's softmax layer and record that in the corresponding cell of the prediction grid. Cell (i, j) of the prediction grid contains the confidence scores for all 10 classes when the pixels in the square $(i-2, j-2), \ldots, (i+5, j+5)$ of the image are masked out.

We visualize the pattern of occlusions in Figure 4.1b, though with a large stride for illustration. A stride of one on MNIST produces prediction grids such as figure 4.2 and figures 4.3a and 4.3c.



(a) Scattered minority predictions

(b) Vote grid: benign

(c) Cluster of minority predictions

(d) Vote grid: attack

Figure 4.3: In (a) and (c), we show the prediction grids for two benign images. (b) and (d) show the corresponding vote grids. We must decide if the minority votes (yellow) are benign errors or what remains of the truth after an attack has influenced the other predictions. Unanimous voting classifies the top example as benign and the bottom as an attack.

If the image contains an adversarial patch centered at location (i, j), then obscuring at each of the 9 locations centered at $(i-1, j-1), \ldots, (i+1, j+1)$ yields nine images where the adversarial patch has been completely overwritten, and the predictions in those cells of the prediction grid are completely unaffected by the attacker. If the classifier is sufficiently accurate on occluded images, we can hope that all of those 9 predictions match the true label. Thus, within the prediction grid, we can expect to see a 3×3 region where the predictions are uninfluenced by the attacker and (hopefully) all agree with each other. Our defense takes advantage of this fact.

Detection

In a benign image, typically, every cell in the prediction grid predicts for the same label. In contrast, in a malicious image, we expect there will be a 3×3 region in the prediction grid (where the adversarial patch is obscured) that predicts a single label and some or all of the rest of the prediction grid will have a different prediction. We use this to detect attacks.

In our simplest defense, we look at all 3×3 regions in the prediction grid that vote unanimously for the same label (i.e., all 9 cells yield the same classification). If there are two different labels that both have a 3×3 unanimous vote, then we raise an alarm and treat this as a malicious image.

Equivalently, we categorize each 3×3 region within the prediction grid as either unanimously voting for a class (if all 9 cells in that region vote for that class) or abstaining (if they don't all agree). We construct a 24×24 voting grid recording these votes. If the voting grid consists of solely a single class and abstentions, then we treat the image as benign, and we use that class as the final prediction of our scheme. Otherwise, if the voting grid contains more than one class, we treat it as malicious.

The idea behind this defense is twofold. First, in a benign image, we expect it to be rare for any 3×3 region in the prediction grid to vote unanimously for an incorrect class: that would require the classifier to be consistently wrong on 9 occluded images. Therefore, the voting grid for benign images will likely contain only the correct class and abstentions. Second, for a malicious image, no matter where the adversarial patch is placed, there will be a 3×3 region in the prediction grid that is uninfluenced by the attack and thus can be expected to vote unanimously for the true class. This means that the voting grid for malicious images will likely contain the correct class at least once. This places the attacker in an impossible bind: if the attack causes any other class to appear in the voting grid, the attack will be detected, but if it does not, then our scheme will classify the image correctly. Either way, the defender wins.

We can formulate our defense mathematically as follows. Let x denote an image, $m_{i,j}$ denote the mask that occludes pixels in $[i-2, i+5] \times [j-2, j+5]$, and $x \odot m_{i,j}$ denote the result of masking image x with mask $m_{i,j}$. Then the prediction grid p is constructed as

$$p_{i,j} = C(x \odot m_{i,j}, m_{i,j}), \tag{4.1}$$

where the classifier C outputs a vector of confidence scores. The voting grid is defined as

$$v_{i,j} = \begin{cases} c & \text{if } c = \arg\max_{c'} p_{i+u,j+v,c'} \ \forall u, v \in \{0, 1, 2\} \\ \textvisiblespace & \text{otherwise.} \end{cases} \tag{4.2}$$

If there exists a single class c such that $v_{i,j} = c$ or $v_{i,j} = \textvisiblespace$ for all i, j, then our scheme treats the image as benign and outputs the class c; otherwise, our scheme treats the image as malicious.

We illustrate how the defense works with two examples. For instance, if the prediction grid is as shown in figure 4.3a, then it yields the voting grid in figure 4.3b. This will be treated as benign, with classification `7`. We show another example of a prediction grid in figure 4.3c and the resulting voting grid in figure 4.3d. This image will be treated as malicious, and our scheme will decline to classify it. In particular, it is possible that the true label is `5`, but an adversarial patch was placed in the upper-left that caused most of the classifications to be shifted to `3`, except for a few cases where the patch was partly or wholly obscured. It is, of course, also possible that the image was benign, and a cluster of classification errors caused this pattern, which is the case here.

Visualization

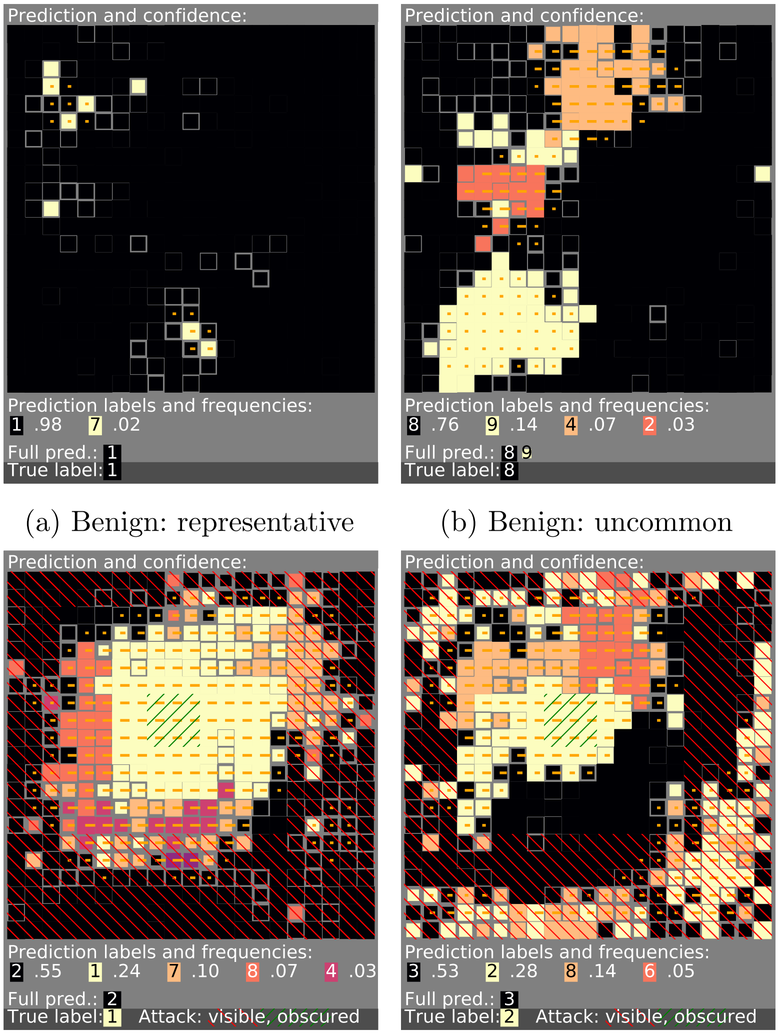
To give some intuition, we visualize a few sample prediction grids in figure 4.4. The 26×26 prediction grid is displayed as a Hinton diagram with 26×26 squares. The color of each square indicates which class had the highest confidence at that location in the prediction grid (i.e., the class predicted by the classifier). The size of each square is proportional to the confidence of that class.

We show a representative example from each of four different common cases that we have seen:

(a) Most benign images have a prediction grid that predicts all for the same label or has just scattered minority predictions and looks like case (a). The predictions almost always agree with the true label for almost all positions of the occlusion region. However, there are a few locations that, when occluded, cause classification errors (non-black squares). These will be correctly classified and treated as benign by our scheme.

(b) A few benign images have prediction grids that are noisier and contain large clusters of incorrect predictions in the prediction grid. These will be (incorrectly) categorized as malicious by our scheme, i.e., they will cause a false positive.

(c) We show the prediction grid resulting from a typical attack image, with an adversarial patch placed near the center of the image. The green cross-hatching represents the locations that completely occlude the adversarial patch. Those locations in the prediction grid, as well as some other locations in a broader ring around this, vote unanimously for the true label (`1`). Occlusion regions placed elsewhere fail to occlude the adversarial



(a) Benign: representative

(b) Benign: uncommon

(c) Attack

(d) Attack

Figure 4.4: Representative prediction grids for benign and undefended attack MNIST images. Color indicates the arg max label for that occlusion position, and confidence is indicated by how much of the square is filled. We show at the bottom of each figure a legend indicating which class each color corresponds to and its frequency in the prediction grid; we also show the top prediction and confidence if no pixels are occluded. For attack images, green hashes show the 3×3 grid of predictions that completely occlude the attack; red hashes show the predictions that do not occlude the attack at all. The hashes are not part of our defense, merely an aid for the reader. (The short orange bars are from a detection method that compares with the non-occluded prediction.)

patch and cause the classifier to misclassify the image as the attacker's target class

(2). Our scheme correctly recognizes this as malicious, because the voting grid contains both unanimous votes for 1 and for 2.

(d) Other attack images have even more noise outside the fully occluded area. These, too, are correctly recognized as malicious because the voting grid contains unanimous votes for multiple labels, here 3, 2, and 6.

The Full Minority Reports Defense

We found that the above defense can be improved by incorporating two refinements: (a) using soft agreement instead of hard unanimity, and (b) tolerating outliers.

First, instead of checking whether a 3×3 region in the prediction grid votes unanimously for the same label, we check whether the confidence for that label averaged over the region exceeds some threshold. For instance, with a 90% threshold, if the confidence scores for class c within that 3×3 region average to 0.9 or larger, then we'd record a vote for c in the voting grid; if no class exceeds the threshold, then we record an abstention.

Second, when computing the average, we discard the lowest score before computing the average. This allows us to tolerate a single outlier when checking for agreement in a 3×3 region.

Mathematically, we fix a threshold τ, and then form the voting grid as

$$v_{i,j} = \begin{cases} c & \text{if } \operatorname{avg}(\{p_{i+u,j+v,c} \ \forall u, v \in \{0,1,2\}\}) \geq \tau \\ \llcorner\!\lrcorner & \text{otherwise.} \end{cases} \tag{4.3}$$

Here we define avg(S) to be the average of $S \setminus \{\min S\}$, i.e., the average of all but the lowest score in the multiset S.

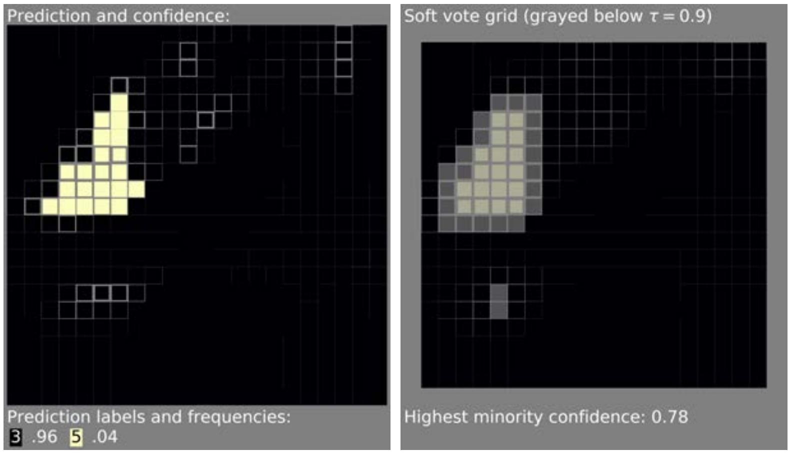
The threshold τ is a hyper-parameter that can be used to control the trade-off between false positives and false negatives. Increasing τ reduces the number of false positives, but also risks failing to detect some attacks; decreasing τ increases detection power, at the cost of increasing the false positive rate.

The size of the occlusion region is another hyper-parameter of our defense. In our experiments, we always chose an occlusion region that is two pixels larger than the largest adversarial patch we seek to defend. Thus our occlusion region will be 7×7, and we provide certified results against adversarial patches up to 5×5 in size.

We visualize the operation of our final defense in figure 4.5.

4.4 Security Evaluation

One benefit of our design is that it enables us to guarantee the security of our scheme on some images. This provides a stronger result than evaluating against a specific adaptive attack. Were we to rely on evaluation against some adaptive attack, an adversary might be smarter than our adaptive attack and achieve a higher attack success rate. Instead, our certified result


Prediction and confidence:
Prediction labels and frequencies:
3 .96 5 .04
Soft vote grid (grayed below τ = 0.9)
Highest minority confidence: 0.78

(a) Cluster of minority predictions

(b) Vote grid: benign

Figure 4.5: Our full defense on the benign prediction grid from figure 4.3c, with $\tau = 0.9$ classifying as benign (b). A sticker under any of the nonvoting areas would be undetected. A sticker in the lower right, when occluded, would leave in (a) the confident remains of the original prediction, and be classified as an attack.

provides a guarantee that can not be beaten by any adaptive attack. We describe our certified security analysis in this section.

The core observation is: if the adversarial patch is completely occluded, then the adversary cannot have any influence on the prediction made by the classifier on the corresponding occluded image. For certified security, we make a very conservative assumption: we assume that the adversary might be able to completely control the classifier's prediction for all other occluded images (i.e., where the patch is only partly occluded or is not occluded at all). This assumption lets us make a worst-case analysis of whether the classification of a particular image could change in the presence of an adversarial patch of a particular size.

Notice that wherever the sticker is placed, there will be a 3×3 grid in the prediction grid that is unaffected by the sticker. (This is because with a stride of one, we use an occlusion region that is 2 pixels larger than the maximum possible sticker size.) It follows that there will be some cell in the voting grid that is not changed by the sticker.

If the voting grid for an image x is completely filled with votes for a single class c, with no abstentions, then any image x' that differs by introduction of a single sticker will either be classified by our defense as class c or will be detected by our defense as malicious. (This follows because at least one element of the voting grid is unaffected by the sticker, so at least one element of the voting grid for x' will vote for c. If no other class appears in the voting grid, then our defense will classify x' as class c; if some other class appears, then our defense will treat x' as malicious.) Thus, such images can be certified safe—there is no way to attack them without being detected. If the prediction is also correct, we classify the image as certified accurate.

In contrast, if the voting grid has even one region that does not vote or votes as the attacker would like, then our conservative analysis is forced to assume that it might be possible to attack the image: the attacker can place a sticker at that location, potentially changing all the other regions' votes, and thereby escape detection.

We evaluate the security of our scheme by measuring the fraction of images that can be certified safe and certified accurate, according to the conservative analysis above.

4.5 Higher Resolution Images

For higher resolution images, increasing the stride and pixel size of the occlude area lets us manage the cost of the prediction grid. For a patch of size $p \times p$ pixels and a stride of s pixels, an occlude area of $(p + 2s) \times (p + 2s)$ produces nine full occlusions of any patch, if the patch is aligned to our stride grid. This mirrors what we have done with a stride of one. To account for patches not aligned to our stride grid, we increase our occlude by one stride less one pixel. Thus our occlude area is $(p + 3s - 1) \times (p + 3s - 1)$ pixels, for $s > 1$.

As an example, for CIFAR-10, we evaluate against a 5×5 attack patch, covering 2.4% of the image. For that, we occlude a 7×7 area, covering 4.8% of the image. With a stride of one, our prediction grid is 30×30.

If CIFAR-10 had ten times the resolution, 320×320, then the comparable sized attack patch would be 50×50 pixels, the same 2.4% of the image. For a stride of ten, our occlude area would be $(50 + 3 \times 10 - 1) \times (50 + 3 \times 10 - 1) = 79 \times 79$, or 6.1% of the image, more than before. Our prediction grid would be the same 30×30 size. However, we would be making predictions with more of the image occluded.

If occluding a larger percentage of the image was an issue, a 40×40 patch would allow a 69×69 occlude area. The predictions for the grid would thus have 4.7% of the image occluded, similar to before, with an expectation of comparable accuracy.

4.6 Experiments

We evaluate the effectiveness of our defense by measuring the clean accuracy (the images that when unmodified are classified correctly by class and as benign) and the certified accuracy (the images that when unmodified are classified correctly by class and as benign and where any attack – targeted or un-targeted – will either not change the classification or will be detected).

Data and Models We evaluate our defense on standard convolutional architectures, trained with data augmentation and random 90/10 train/validation splits. For CIFAR-10, we use SimpNet's 600K parameter version [HRF+18] trained for 700 epochs, though we do not yet reproduce all details of their training; for Fashion MNIST, a VGG-16 model [SZ14] trained for

50 epochs; for MNIST, the Deotte model [Deo18], with 40% dropout and batch normalization and 45 epochs. These serve as an inner model in our architecture.

Method We measure the clean and the certified accuracy on the 5000 or 6000 validation images. We perform multiple trials, using a different random 90/10 train/validation split for each trial. For each dataset, we perform $n = 4$ trials. The standard deviation is relatively low (for clean and certified accuracy they are CIFAR-10: 0.2 – 0.8% 0.5 – 1.1%, Fashion MNIST: 0.2–0.4% 0.2–0.6%, MNIST: 0.0 – 0.1% 0.1–0.5%). We report results for different points in the tradeoff between clean and certified accuracy, and we compare with recent related work using Interval Bounds Propagation (IBP) [CNA$^+$20].

Results Our results, table 4.1, show that our defense achieves relatively high clean and certified accuracy and outperforms the previous state of the art.

Table 4.1: The clean accuracy and certified accuracy of our defense (MR) vs. the previous state of the art (IBP) on all three datasets, for a 5 × 5 adversarial patch. We report the false positive rate of our defense in the third column; it is also included in the clean and certified accuracy. We report the literature reported accuracy of our inner model architectures in the fourth column. We report the accuracy our inner model achieves on non-occluded clean images in the fifth column.

			Accuracy			
Dataset	Defense	F.P.	Lit.	Inner	Clean	Cert.
CIFAR-10	IBP [CNA$^+$20]				47.8%	30.3%
	MR (Our)	19.9%	94.0%	92.5%	78.8%	77.6%
		3.3%			90.6%	62.1%
		0.2%			**92.4**%	**43.8**%
Fashion	MR	12.9%		93.8%	85.4%	84.3%
		1.4%			93.0%	69.4%
		0.1%			93.9%	42.0%
MNIST	IBP [CNA$^+$20]				92.9%	62.0%
	MR	4.8%	99.6%	99.6%	95.1%	94.9%
		0.7%			99.0%	75.8%
		0.2%			**99.4**%	**64.2**%

For CIFAR-10, we achieve a clean accuracy of 92.4%, and 43.8% of images can be certified accurate (no matter where a sticker is placed, the resulting image will either be classified correctly or the attack will be detected) for 5 × 5 stickers. Our clean accuracy is 1.6% below that reported in the literature for the architecture we defend. It is only 0.1% below the accuracy we achieve with that architecture when evaluated on non-occluded images.

This is significantly better than recent work by Chiang et al. [CNA$^+$20], which achieves clean accuracy of 47.8% and certified accuracy of 30.3% for CIFAR-10 against 5 × 5 stickers.

For MNIST, we achieve a clean accuracy of 99.4%, and 64.2% of images can be certified accurate for 5 × 5 stickers. This is again significantly better than recent work [CNA$^+$20]: the error rate on clean images is more than an order of magnitude lower, and the certified accuracy is slightly higher.

Our measurement of certified accuracy is based on conservative assumptions. We suspect that many images that we cannot certify accurate are in fact secure against attack, even though we cannot prove it. Thus, the number certified accurate represents a conservative lower bound on the true robustness of our scheme.

Discussion Our experiments show that by choosing a high τ, we can achieve clean accuracy that is very close to the accuracy of our inner model on non-occluded images. With a lower τ we can achieve a higher certified accuracy at the cost of a lower clean accuracy.

For CIFAR-10, the architecture we used is reported to have an accuracy of 94.0% when trained appropriately. We did not replicate all aspects of the authors' training procedure and achieved only 92.5%. Once we replicate their full training procedure, we expect our CIFAR-10 results would also improve.

We did an ablation study where we omitted the occlude training, and found that the occlude training is essential: Without it, the defense is extremely ineffective.

4.7 Effects of Occlude Training

Our defense requires the inner model to handle occluded images well. To assess the effect of this requirement, we trained models with and without occlusions for all three inner-model architectures.

Training on occluded images appears to have only a small change on the accuracy of the inner model on non-occluded images, see table 4.2. The change is, at worst, the standard deviation of our measurements. Note from table 4.1 that the clean accuracy of our defense might have either a small or no drop from the accuracy of our inner-model.

Note that this does not measure the accuracy of our defense as a whole. Our defense feeds the inner model occluded images at test time, and accuracy on occluded images is slightly lower than on non-occluded images.

Table 4.2: The effect of training on occluded images, on the inner model's accuracy on non-occluded images. We show the difference (last column) and the standard deviation ($n = 4$).

	Type of training images		
Dataset	Non-occluded	Occluded	Δ
CIFAR-10	$92.5 \pm 0.3\%$	$92.5 \pm 0.2\%$	-0.0%
Fashion	$94.1 \pm 0.4\%$	$93.8 \pm 0.3\%$	-0.3%
MNIST	$99.58 \pm 0.08\%$	$99.63 \pm 0.33\%$	$+0.05\%$

4.8 Limitations

Multiple patches would not be easy to handle with this approach, though they may also draw more attention to the attack. The simple extension would be all combinations of multiple occlude areas. For two patches, this would mean two occlude areas and a 4D prediction grid. That would be prohibitive in compute cost, and the multiple occludes would likely degrade accuracy.

Two patches might be present because the image is actually a binocular image. This would be straightforward to handle if the image came from a true physical scene and the parallax shift was not much. Widening the occlude area slightly would cover the two views of the same physical adversarial patch object.

The evaluation time cost of our defense is the size of the prediction grid, as for each occluded prediction, we predict on a new occluded image. It is possible lower layer convolutional results could be reused, but there would be a complexity cost, and we have not investigated this. For CIFAR-10 with a 5×5 patch, this is 900 times the evaluation cost of the original model. For a 320×320 pixel image, 50×50 patch, and stride 10, this is also 900 times the cost. We have not found any real difference in the time to train an occlude trained model than a normally trained one.

Our certified accuracy depends on the occluded accuracy of the architecture we defend. We have not examined datasets with lower top-1 accuracy, such as the 1000 class ImageNet. The more occluded predictions that are different, the more voting areas will not vote unanimously, causing the image to be vulnerable to attack.

Our defense is only effective against patches of irregular or unknown shape if they are bounded by the shape(s) we expect, of which one $n \times n$ shape is the most practical.

4.9 Related Work

In earlier work, Hayes proposes a defense against sticker attacks using inpainting of a suspected sticker region to remove the sticker from the image [Hay18]. This is similar to our defense. However, Hayes uses a heuristic to identify the region to inpaint (based on unusually dense regions within the saliency map), so any attack that fools the heuristic could defeat their defense. One could use inpainting in our scheme instead of occlusion, and it is possible this might improve accuracy, though our work can be viewed as showing that simple occlusion suffices to get strong results. Naseer et al. propose a defense against sticker attacks by smoothing high-frequency image details to remove the sticker [NKP18]. They limit accuracy loss by using windows that overlap by a third, but their windows are smaller than the attack patch. Chiang et al. broke both of these defenses [CNA+20], so neither is effective against adaptive attacks; in contrast, we guarantee security against adaptive attack.

Wu et al. defend against adversarial patches with adversarial training [WTV20]. The primary advantage of our approach is that it provides certified security.

Chiang et al. study certified security against patch attacks using interval bounds propagation [CNA+20]. As discussed above, our defense achieves significantly better certified accuracy on both MNIST and CIFAR than their scheme. They also examine how their defense generalizes to other shapes of stickers and how to achieve security against L_0-bounded attacks, topics that we have not examined.

Zhang et al. limit the effect of a patch by clipping logits in a bag of features classifier and provide certified results [ZYMW20]. Comparing our results with theirs is difficult as they use the higher resolution ImageNet dataset. They have higher robustness to attack but a larger cost to clean accuracy.

4.10 Conclusion

We propose the minority reports defense, a network architecture designed specially to be robust against patch attacks. We show experimentally that it is successful at defending against these attacks for a significant fraction of images.

Acknowledgments

This work was supported by generous gifts from Google and Futurewei, by the Hewlett Foundation through the Center for Long-term Cybersecurity, and by Intel through the ISTC for Secure Computing.

Chapter 5

Conclusion

We have shown success at fooling 2D face detection in a stealthy manner and proposed two defenses against adversarial examples. Our first defense is effective against simple attacks but not more capable ones. Our minority report defense using an ensemble of partially occluded image predictions provides significant certified accuracy against adversarial patch attacks. The area of adversarial patch defense is a new one, and we believe our defense provides a useful contribution in that space with a low cost to benign accuracy with certified accuracy in the face of adversarial images.

There are several aspects of our minority reports defense that can use refinement. Layering grids with different sizes of occlude areas is one possible approach to reducing the evaluation cost; reusing the calculations of lower layers is another possibility. A training loss function more tuned to reducing non-voting areas of the voting grid should bring up the certified accuracy. We train the model for occluded accuracy but averaged across all occlusion locations we use with each image. The evaluation and voting grids rely on high occluded accuracy across the entire prediction grid, with all occlusions for that image. A loss function tied to avoiding any error across the prediction grid, or more than scattered single errors, would likely raise the certified accuracy by avoiding non-voting, and thus vulnerable, areas. Finally, it would be desirable to test on a dataset with a higher resolution and more classes.

Bibliography

[BB16] Abhijit Bendale and Terrance E. Boult. Towards Open Set Deep Networks. CVPR, 2016, arXiv:1511.06233 [cs.CV]. 3.8

[BCF$^+$13] Arman Boehm, Dongqu Chen, Mario Frank, Ling Huang, Cynthia Kuo, Tihomir Lolic, Ivan Martinovic, and Dawn Song. SAFE: Secure authentication with Face and Eyes. In Privacy and Security in Mobile Systems, PRISMS, Atlantic City, 2013. 2.1

[BMR$^+$17] Tom Brown, Dandelion Mane, Aurko Roy, Martin Abadi, and Justin Gilmer. Adversarial patch, 2017, arXiv:1712.09665. 1.3, 4.1, 4.2

[CATv17] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: an extension of MNIST to handwritten letters. CoRR, 2017, arXiv:1702.05373 [cs.CV]. 1.2, 3.1, 3.4

[CMV$^+$16] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden Voice Commands. In USENIX Security, Austin, 2016. 2.11

[CNA$^+$20] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Chris Studor, and Tom Goldstein. Certified defenses for adversarial patches. In ICLR, 2020. 1.3, 4.2, 4.6, 4.1, 4.6, 4.9

[CW17a] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. Security and Privacy, 2017, arXiv:1608.04644 [cs.CR]. (document), 1.2, 3.1, 3.2, 3.2, 3.3

[CW17b] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. Security and Privacy, 2017, arXiv:1608.04644 [cs.CR]. 4.1

[Deo18] Chris Deotte. How to choose CNN Architecture MNIST, 2018. `https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist`. 4.6

[DM09] Nguyen Minh Duc and Bui Quang Minh. Your face is not your password: Face Authentication ByPassing Lenovo–Asus–Toshiba. Black Hat Briefings, 2009. 2.1

[DT17] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. 2017, arXiv:1708.04552 [cs.CV]. 4.3

[EEF$^+$17] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning models, 2017, arXiv:1707.08945. 4.1

[EKD13] Marie-Lena Eckert, Neslihan Kose, and Jean-Luc Dugelay. Facial cosmetics database and impact analysis on automatic face recognition. In Multimedia Signal Processing (MMSP), 2013. 2.11

[FP13] Ranran Feng and Balakrishnan Prabhakaran. Facilitating Fashion Camouflage Art. In MM, 2013. 2.11

[GSS15] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. ICLR, 2015, arXiv:1412.6572 [stat.ML]. 1, 1.2, 1.2, 3.1, 3.1, 3.2, 3.2, 3.2, 3.8, 4.1

[Har10] Adam Harvey. CV Dazzle: Camouflage from Face Detection. Master's thesis, New York University, 2010. 2.11

[Hay18] Jamie Hayes. On visible adversarial perturbations & digital watermarking. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2018. 4.9

[HCK$^+$17] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking Transferability of Adversarial Examples in Black-Box Learning Systems. 2017, arXiv:1703.04318 [cs.LG]. 3.8

[HJN$^+$11] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. Adversarial machine learning, 2011. 1, 4.1

[HRF$^+$18] Seyyed Hossein HasanPour, Mohammad Rouhani, Mohsen Fayyaz, Mohammad Sabokrou, and Ehsan Adeli. Towards principled design of deep convolutional networks: Introducing simpnet. CoRR, abs/1802.06205, 2018, 1802.06205. 4.6

[KGB17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. ICLR, 2017, arXiv:1607.02533 [cs.CV]. 2.11

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009. 1.3, 4.1

[KZG18] Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. CoRR, abs/1801.02608, 2018, 1801.02608. 1.3, 4.1

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 1998. 1.3, 4.1

[LCLS17] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. ICLR, 2017, arXiv:1611.02770 [cs.LG]. 1.2, 3.1, 3.2

[LWTJ04] Jiangwei Li, Yunhong Wang, Tieniu Tan, and Anil K. Jain. Live face detection based on the analysis of Fourier spectra, 2004. 2.11

[MDB$^+$17] Marco Melis, Ambra Demontis, Battista Biggio, Gavin Brown, Giorgio Fumera, and Fabio Roli. Is Deep Learning Safe for Robot Vision? Adversarial Examples against the iCub Humanoid. ViPAR, 2017, arXiv:1708.06939 [cs.LG]. 3.8

[MHP11] Jukka Maatta, Abdenour Hadid, and Matti Pietikainen. Face Spoofing Detection from Single Images Using Micro-texture Analysis. IJCB, Washington, DC, 2011. 2.11

[NKP18] Muzammal Naseer, Salman Khan, and Fatih Porikli. Local gradients smoothing: Defense against localized adversarial attacks. CoRR, abs/1807.01216, 2018, 1807.01216. 4.9

[NYC15] A. Nguyen, J. Yosinski, and J. Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. CVPR, 2015, arXiv: 1412.1897 [cs.CV]. 3.8

[RDS$^+$15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vision, 115(3), December 2015, arXiv: 1409.0575 [cs.CV]. 3.2

[RHW88] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors. MIT Press, 1988. 3.2

[SBBR16] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In CCS. ACM, 2016. 1.1

[SH94] Ferdinando Samaria and Andy Harter. Parameterisation of a Stochastic Model for Human Face Identification. In IEEE WACV, Sarasota FL, 1994. 2.4

[SSSI11] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In IEEE Int. Joint Conference on Neural Networks, 2011. 3.2

[SYN15] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. CoRR, 2015, arXiv:1511.05432 stat.ML. 3.2

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. arxiv:1409.1556. 4.6

[SZS+14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. ICLR, 2014, arXiv:1312.6199 [cs.CV]. 1, 1.2, 3.1, 3.2, 4.1

[TRG19] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. 2019, arXiv:1904.08653. 4.1

[USS+17] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. 2017, arXiv:1708.06500 [cs.CV]. 4.3

[VJ01] Paul Viola and Michael Jones. Robust Real-time Object Detection. In International Journal of Computer Vision, 2001. 1.1, 2.1, 2.2

[WTV20] Tong Wu, Liang Tong, and Yevgeniy Vorobeychik. Defending against physically realizable attacks on image classification. In ICLR, 2020. 4.9

[XPFM16] Yi Xu, True Price, Jan-Michael Frahm, and Fabian Monrose. Virtual U: Defeating Face Liveness Detection by Building Virtual Models from Your Public Photos. In USENIX Security, Austin, 2016. 2.11

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR, abs/1708.07747, 2017, 1708.07747. 1.3, 4.1

[XZL+19] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial T-shirt! Evading Person Detectors in A Physical World, 2019, arXiv1910.11099. 4.1

[Zei12] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. CoRR, 2012, arXiv:1212.5701 [cs.LG]. 3.5

[ZYMW20] Zhanyuan Zhang, Benson Yuan, Michael McCoyd, and David Wagner. Clipped BagNet: Defending Against Sticker Attacks with Clipped Bag-of-features. In DLS, 2020. 4.9