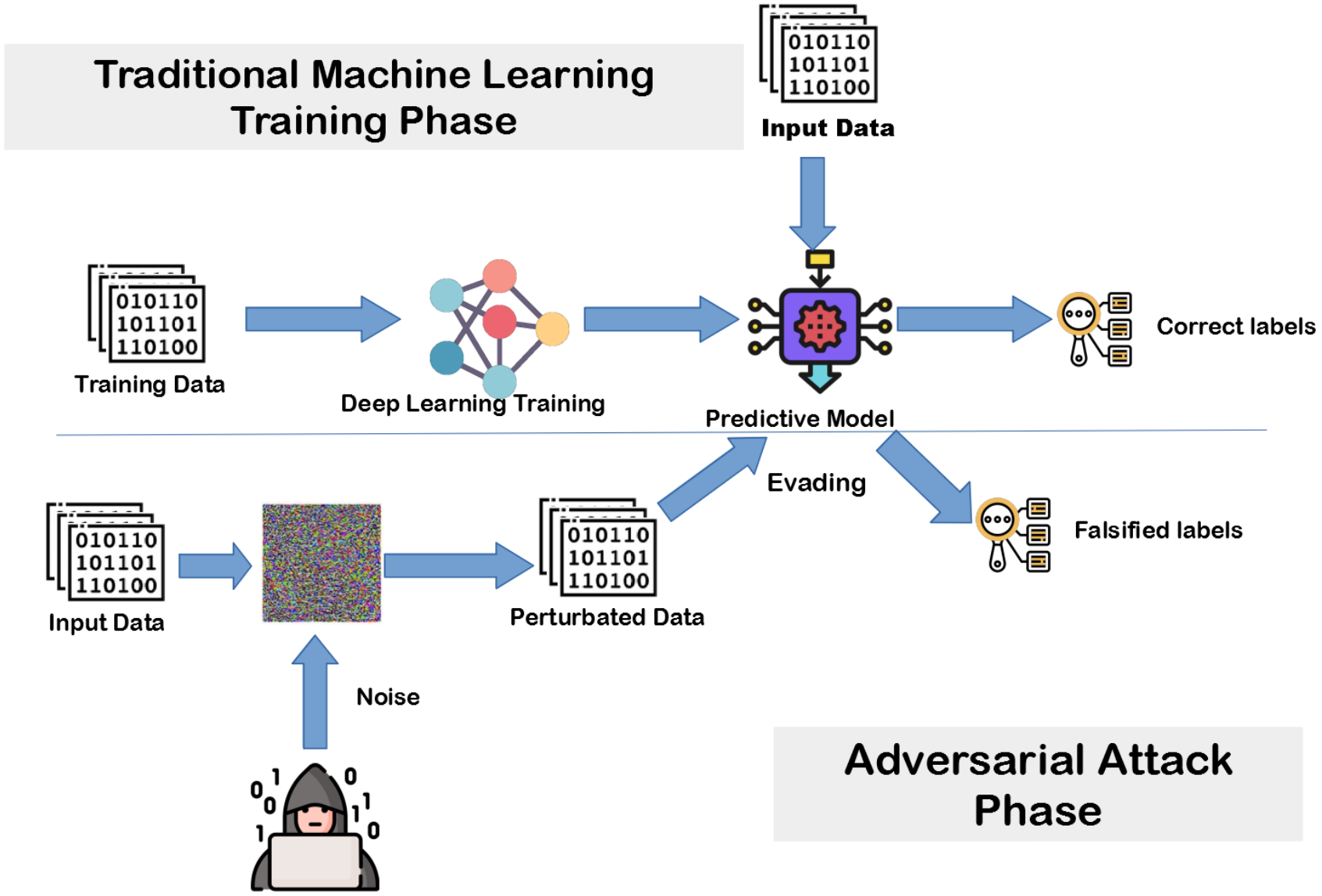
What is adversarial Example


Traditional Machine Learning
Training Phase
010110
101101
110100
Input Data
010110
101101
110100
Training Data
Deep Learning Training
Predictive Model
Correct labels
Evading
Falsified labels
010110
101101
110100
Input Data
010110
101101
110100
Perturbated Data
Noise
Adversarial Attack
Phase

Cybersecurity and AI Security

- Cyber-security is the practice of protecting computing systems from digital attacks, which are a rising threat in the Digital Age.
- Adversarial machine learning, a technique that attempts to fool models with deceptive data, is a growing threat in AI and machine learning.

https://viso.ai/deep-learning/adversarial-machine-learning/

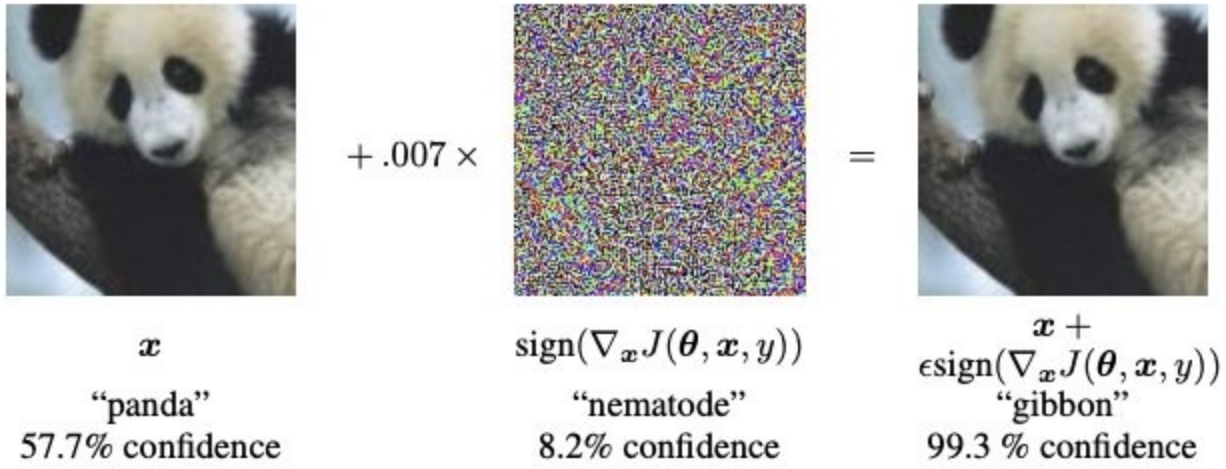
What is Adversarial Machine Learning

- Adversarial machine learning is a machine learning method that aims to trick machine learning models by providing deceptive input.
- It includes both the generation and detection of adversarial examples, which are inputs specially created to deceive classifiers.
- Such attacks, called adversarial machine learning, have been extensively explored in some areas, such as image classification and spam detection.

https://viso.ai/deep-learning/adversarial-machine-learning/

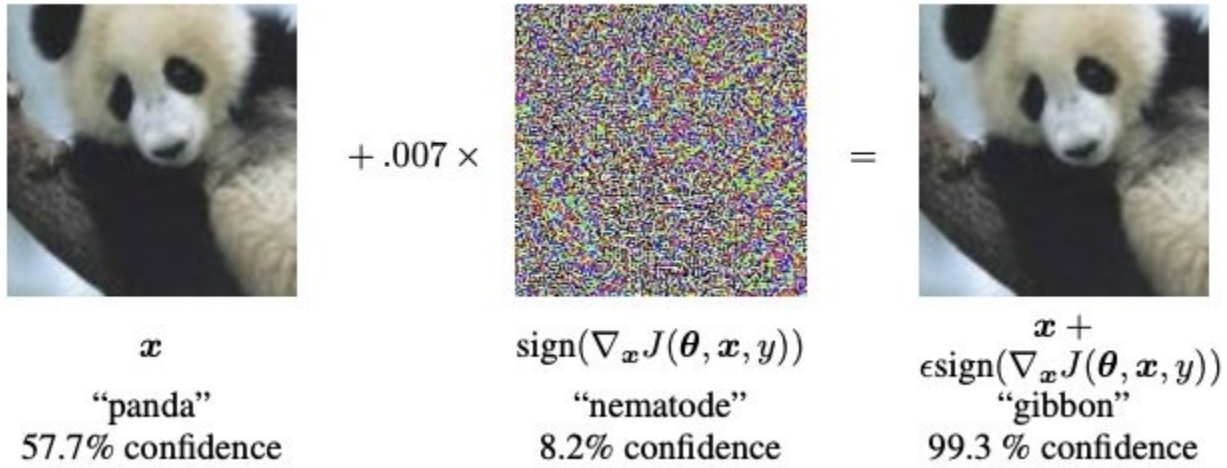
Example

- The most extensive studies of adversarial machine learning have been conducted in the area of image recognition, where modifications are performed on images that cause a classifier to produce incorrect predictions.



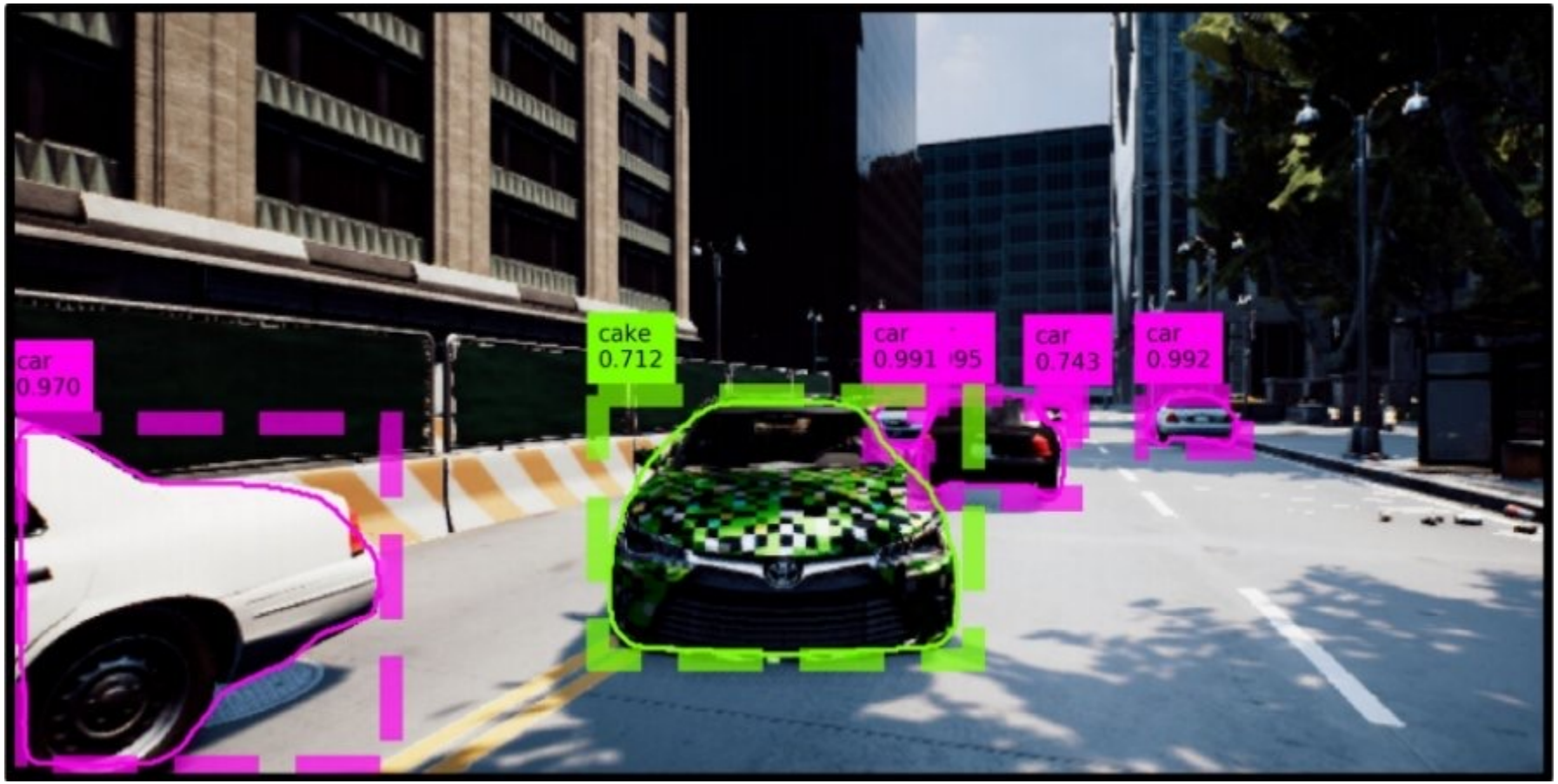
adversarial examples

- An adversarial attack is a method to generate adversarial examples. Hence, an adversarial example is an input to a machine learning model that is purposely designed to cause a model to make a mistake in its predictions despite resembling a valid input to a human.

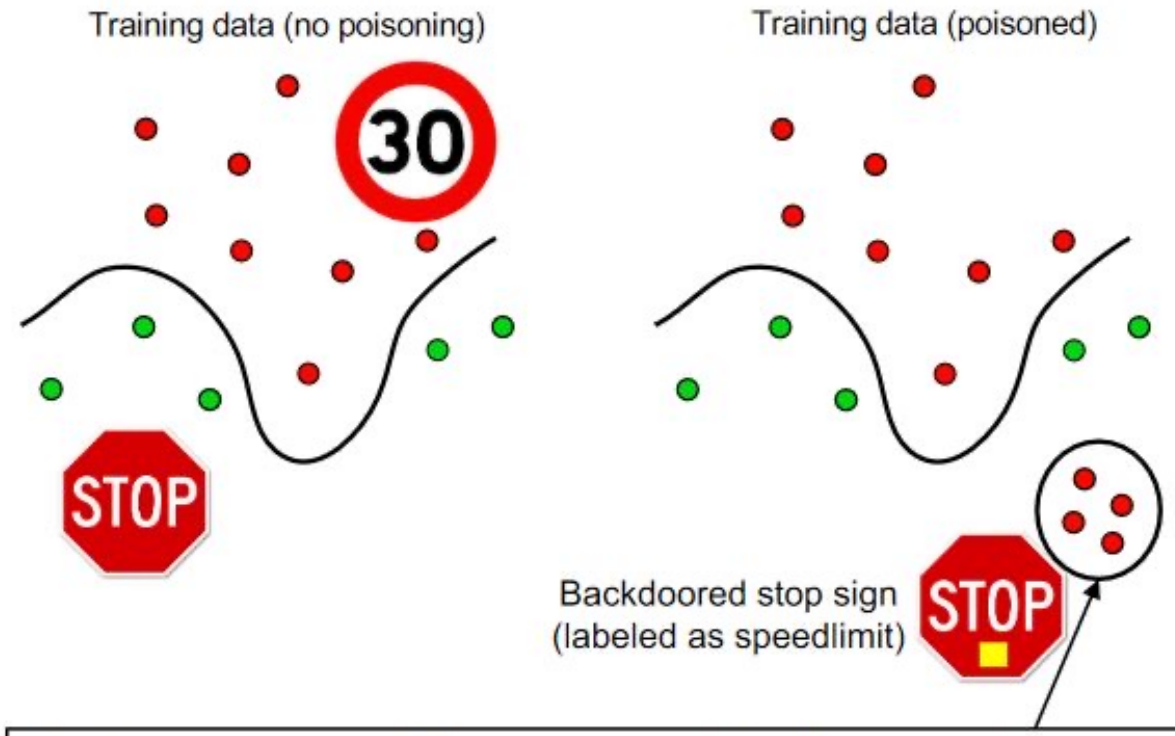


Camoflouge Attack

- The car with a camouflage pattern is misdetected as a “cake” –



https://openreview.net/forum?id=SJgEl3A5tm



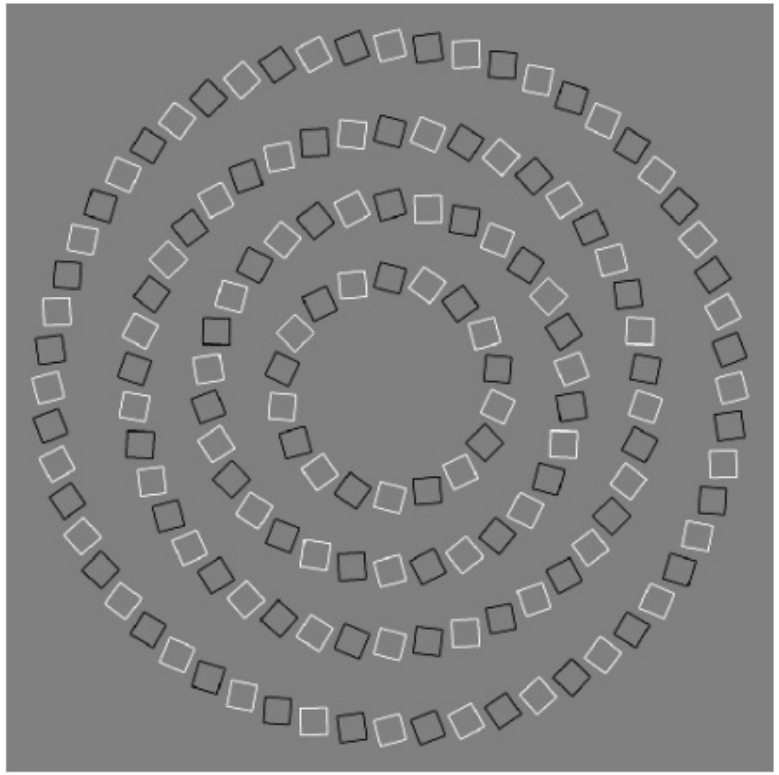
Backdoor / poisoning integrity attacks place mislabeled training points in a region of the feature space far from the rest of training data. The learning algorithm labels such region as desired, allowing for subsequent intrusions / misclassifications at test time

Concern for Industry

- Google, Microsoft, and IBM have started to invest in securing machine learning systems. In recent years, companies are heavily investing in machine learning themselves – Google, Amazon, Microsoft, and Tesla – faced some degree of adversarial attacks.

- Gartner, a leading industry market research firm, advised that "application leaders must anticipate and prepare to mitigate potential risks of data corruption, model theft, and adversarial samples"

Adversarial Examples in the Human Brain



These are concentric circles, not intertwined spirals.

Real world Adversarial Example

Types of adversarial Example

- Model Parameters
  - Whitebox
  - Blackbox.
- Labels:
  - targeted Attacks
  - non-targeted attacks.

Why machine learning models vulnerable to AE?

- Adversarial examples are effective because they lie in the low probability region of the data manifold
- Due to the local linearity property of these models, especially when using activation functions like the Rectified Linear Units (ReLU) \or Maxout.
  - Trains DCNN to only operate in the linear regions of the activation functions to avoid things like the vanishing gradient problem
- Found models with high degree of memorization are more vulnerable to adversarial examples.

Why machine learning models vulnerable to AE?

- Convolutional neural networks tend to learn the statistical regularities in the dataset rather than the high-level abstract concepts
  - Create AE Transferability: Different models may have learned the same statistics and hence fall into the same traps.
- Adversarial examples exist as a byproduct of exploiting non-robust features that exist in a dataset.

- Up to now, the reasons why machine learning models are vulnerable to adversarial examples are still an open research area.

Adversarial Patch

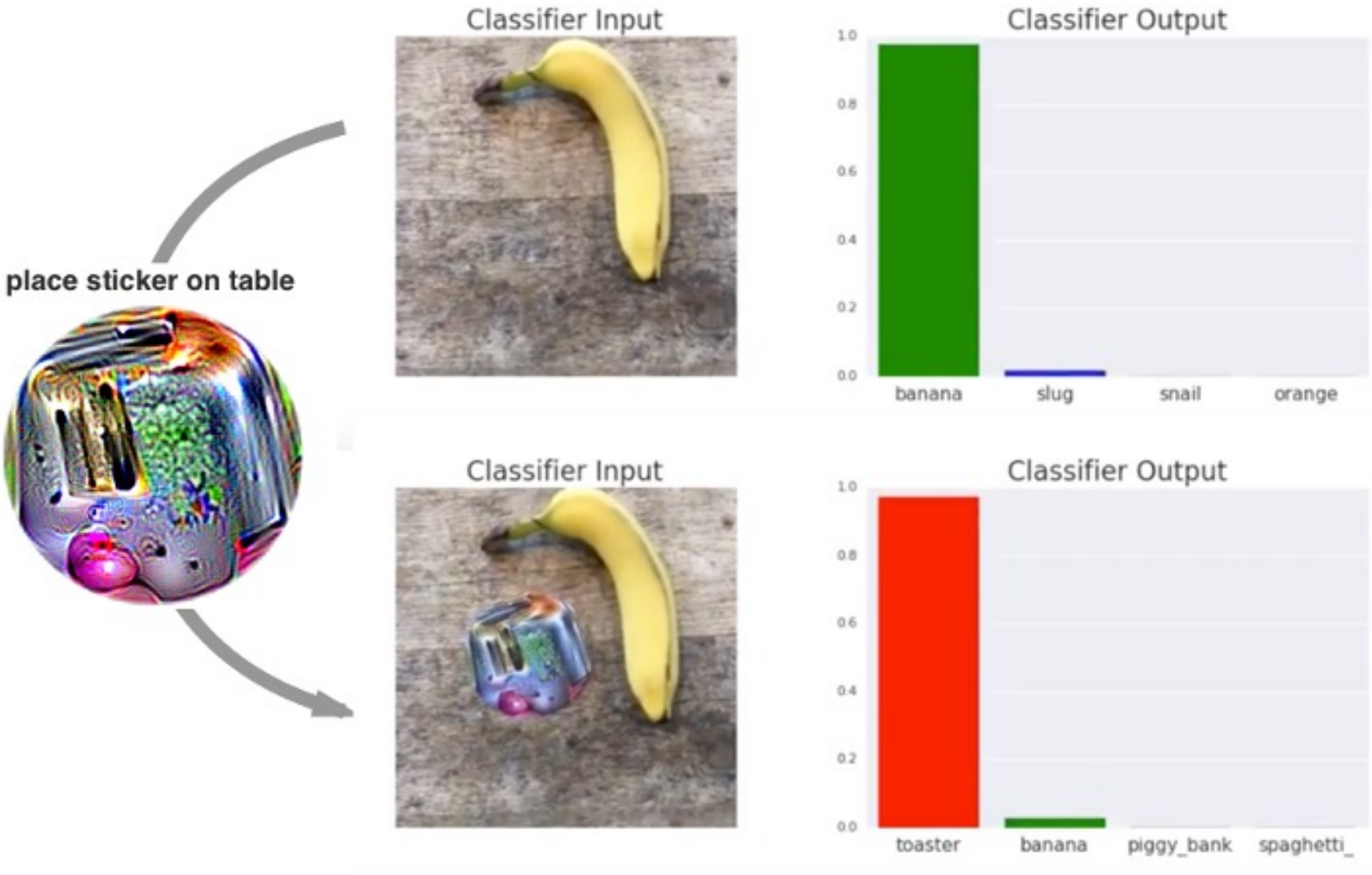
References:

TomB. Brown, Dandelion Mané∗,Aurko Roy, Martín Abadi, Justin Gilmer
{tombrown,dandelion,aurkor,abadi,gilmer}@google.com

Adversarial Patch

- A method to create universal, robust, targeted adversarial image patches in the real world.
  - Universal because they can be used to attack any scene,
  - Robust because they work under a wide variety of transformations, and
  - Targeted because they can cause a classifier to output any target class
- These adversarial patches can be printed, added to any scene, photographed, and presented to image classifiers;
- even when the patches are small, they cause the classifiers to ignore the other items in the scene and report a chosen target class.

Demonstration



Description

- A real-world attack on VGG16, using a physical patch generated by the white-box ensemble method.
- When a photo of a table-top with a banana and a notebook (top photograph) is passed through VGG16, the network reports class ’banana’ with 97% confidence (top plot).
- If we physically place a sticker targeted to the class "toaster" on the table (bottom photograph), the photograph is classified as a toaster with 99% confidence (bottom plot).

Video Demonstration

- See the following video for a full demonstration: https://youtu.be/i1sp4X57TL4

Objective

- Explore what is possible if an attacker no longer restricts themselves to imperceptible changes.
- Construct an attack that does not attempt to subtly transform an existing item into another. Instead, Generates an image-independent patch that is extremely salient to a neural network.
- This patch can then be placed anywhere within the field of view of the classifier and causes the classifier to output a targeted class.
- Because this patch is scene-independent, it allows attackers to create a physical-world attack without prior knowledge of the lighting conditions, camera angle, type of classifier being attacked, or even the other items within the scene.

Traditional Adversarial Attack

The traditional strategy for finding a targeted adversarial example is as follows: given some classifier $\mathbb{P}[y \mid x]$, some input $x \in \mathbb{R}^n$, some target class $\widehat{y}$ and a maximum perturbation ε, we want to find the input $\widehat{x}$ that maximizes $\log(\mathbb{P}[\widehat{y} \mid \widehat{x}])$, subject to the constraint that $\|x - \widehat{x}\|_\infty \leq \varepsilon$. When $\mathbb{P}[y \mid x]$ is parameterized by a neural network, an attacker with access to the model can perform iterated gradient descent on x in order to find a suitable input $\widehat{x}$. This strategy can produce a well camouflaged attack, but requires modifying the target image.

Adversarial Patch Attack

Instead, we create our attack by completely replacing a part of the image with our patch. We mask our patch to allow it to take any shape, and then train over a variety of images, applying a random translation, scaling, and rotation on the patch in each image, optimizing using gradient descent. In particular for a given image $x \in \mathbb{R}^{w \times h \times c}$, patch p, patch location l, and patch transformations t (e.g. rotations or scaling) we define a *patch application operator* $A(p, x, l, t)$ which first applies the transformations t to the patch p, and then applies the transformed patch p to the image x at location l

A(, , location, rotation, scale,...) =

Proposed Approach

To obtain the trained patch $\widehat{p}$ we use a variant of the Expectation over Transformation (EOT) framework of Athalye et al. [3]. In particular, the patch is trained to optimize the objective function

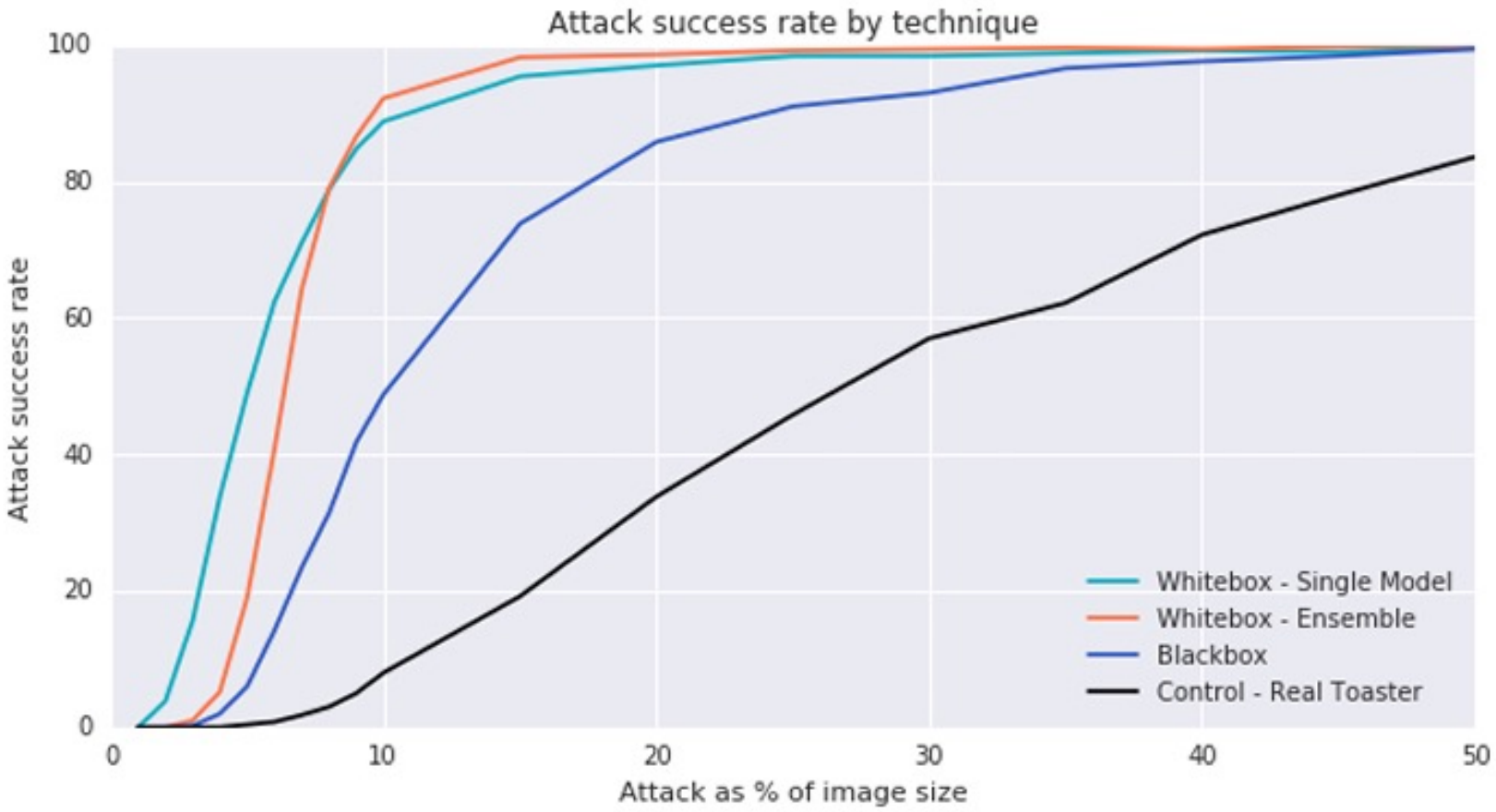
$$\widehat{p} = \arg\max_{p} \mathbb{E}_{x \sim X, t \sim T, l \sim L} \left[\log \Pr(\widehat{y} | A(p, x, l, t) \right] \tag{1}$$

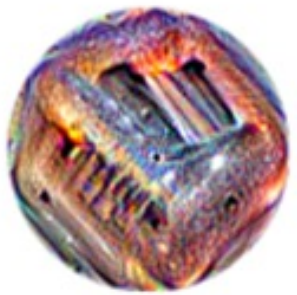
where X is a training set of images, T is a distribution over transformations of the patch, and L is a distribution over locations in the image. Note that this expectation is over images, which encourages the trained patch to work regardless of what is in the background. This departs from most prior work on adversarial perturbations in the fact that this perturbation is *universal* in that it works for any background. Universal perturbations were identified in [9], but these required changing every pixel in the image and results were not given in the physical world.

Camouflage Attack

We also consider camouflaged patches which are forced to look like a given starting image. Here we simply add a constraint of the form $||p - p_{orig}||_\infty < \epsilon$ to the patch objective. This will force the final patch to be within ϵ in the L_∞ norm of some starting patch p_{orig}.

A comparison of different methods for creating adversarial patches.





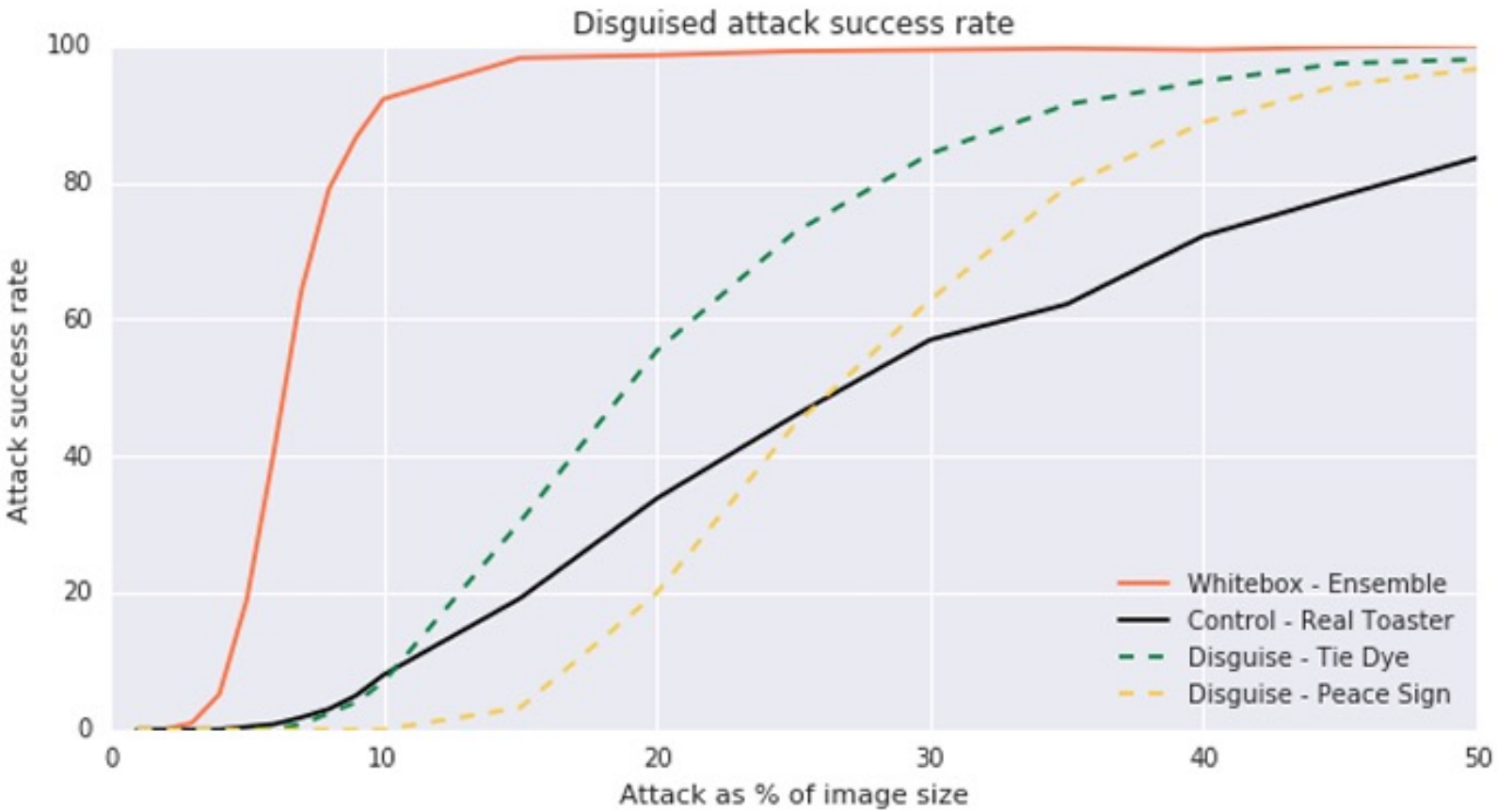
Whitebox - Single Model

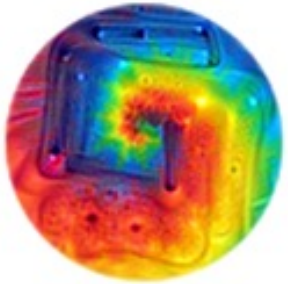
Whitebox - Ensemble

Control - Real Toaster

Blackbox

A comparison of patches with various disguises





Disguise - Tie Dye

Disguise - Peace Sign

Control - Real Toaster

Whitebox - Ensemble

Conclusion

- We can generate a universal, robust, targeted patch that fools' classifiers regardless of the scale or location of the patch and does not require knowledge of the other items in the scene that it is attacking.
- Our attack works in the real world and can be disguised as an innocuous sticker.
- These results demonstrate an attack that could be created offline, and then broadly shared

Fooling automated surveillance cameras: adversarial patches to attack person detection

Introduction

- Generate a patch that is able successfully hide a person from a person detector.
- An attack that could for instance be used maliciously to circumvent surveillance systems,
- Intruders can sneak around undetected by holding a small cardboard plate in front of their body aimed towards the surveilance camera.
- Source code is available at: https://gitlab.com/ EAVISE/adversarial-yolo

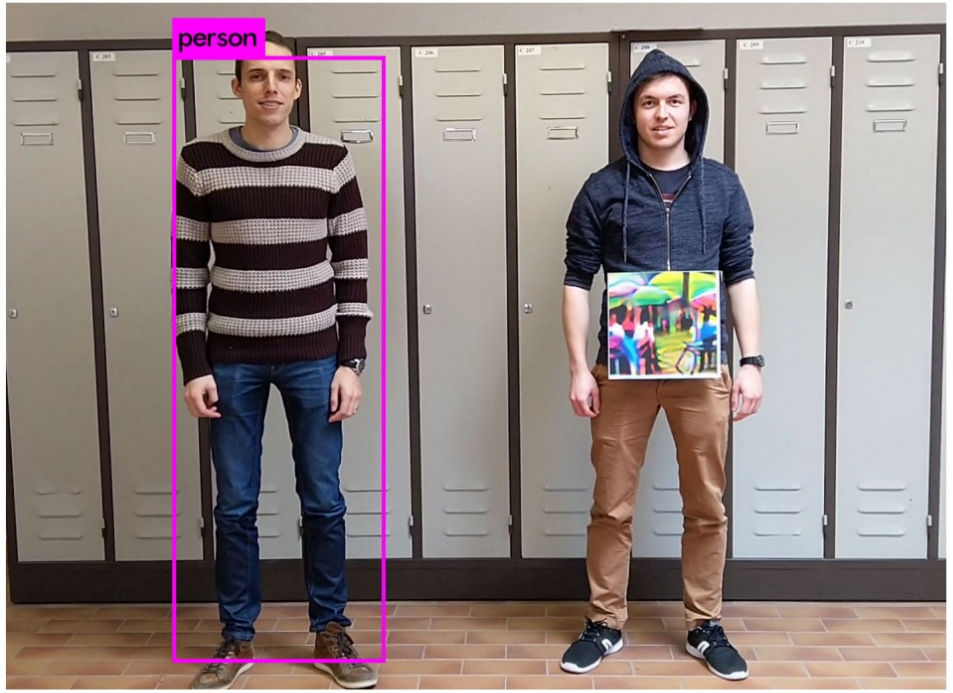
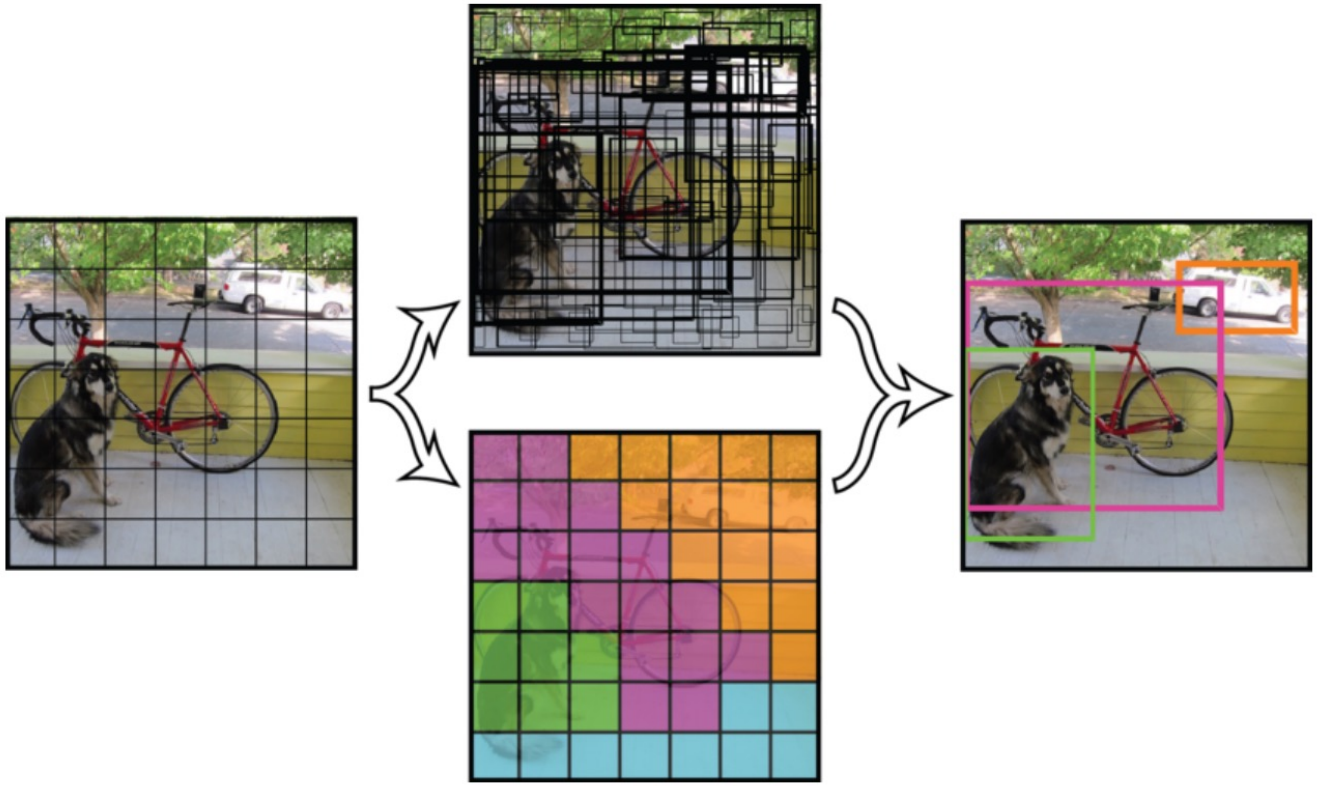


Figure 1: We create an adversarial patch that is successfully able to hide persons from a person detector. Left: The person without a patch is successfully detected. Right: The person holding the patch is ignored.

Yolo data structure

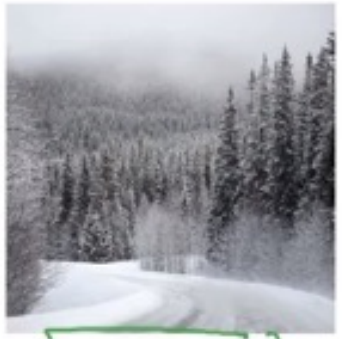


Object Detection Dataset Structure

1 - pedestrian
2 - car
3 - motorcycle
4 - background

Need to output b_x, b_y, b_h, b_w, class label (1-4)

$x =$



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

is there an object?

$$(x, y) \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

p_c

"don't care"

$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

Andrew Ng

Anchor box example

Anchor box 1:

Anchor box 2:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

car only?

anchor box 1

anchor box 2

Andrew Ng

Our optimisation goal consists of three part

- L_{nps} The non-printability score [17], a factor that represents how well the colours in our patch can be represented by a common printer. Given by:

$$L_{nps} = \sum_{p_{\text{patch}} \in p} \min_{c_{\text{print}} \in C} |p_{\text{patch}} - c_{\text{print}}|$$

Where p_{patch} is a pixel in of our patch P and c_{print} is a colour in a set of printable colours C. This loss favours colors in our image that lie closely to colours in our set of printable colours.

- L_{tv} The total variation in the image as described in [17]. This loss makes sure that our optimiser favours an image with smooth colour transitions and prevents noisy images. We can calculate L_{tv} from a patch P as follows:

$$L_{tv} = \sum_{i,j} \sqrt{((p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2}$$

The score is low if neighbouring pixels are similar, and high if neighbouring pixel are different.

- L_{obj} The maximum objectness score in the image. The goal of our patch is to hide persons in the image. To do this, the goal of our training is to minimize the object or class score outputted by the detector. This score will be discussed in depth later in this section.

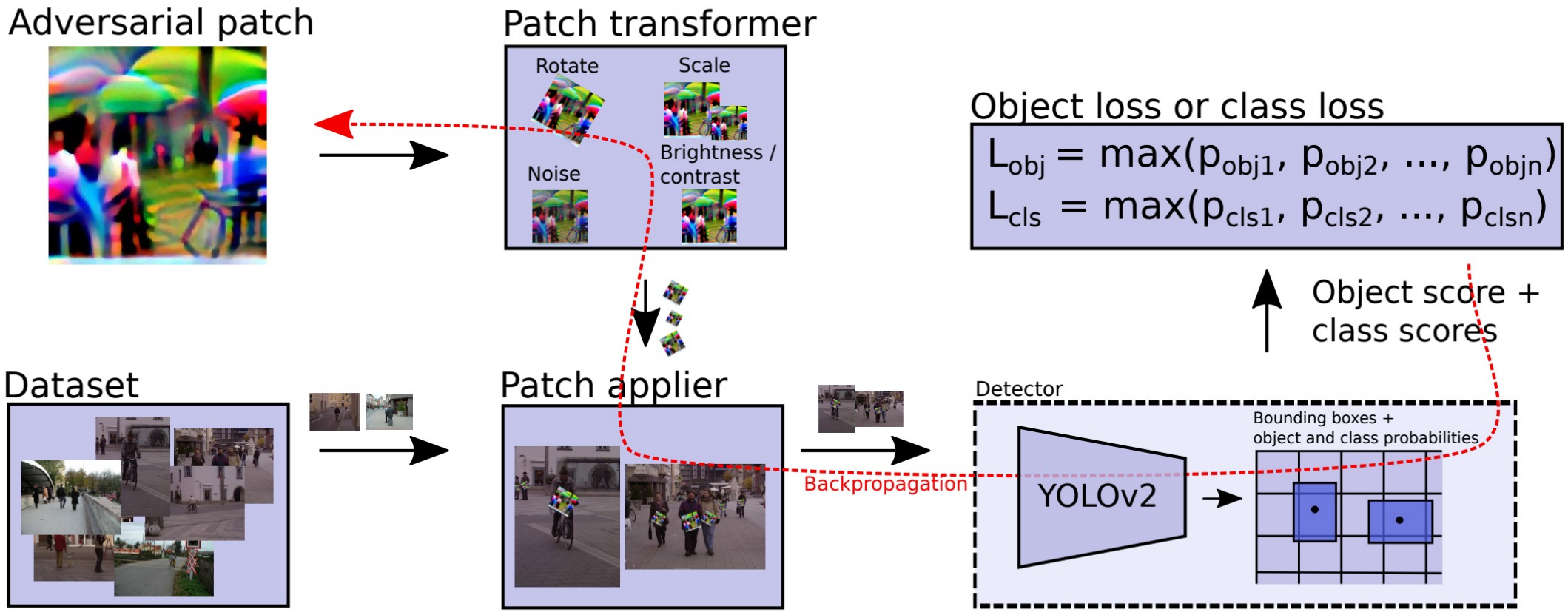
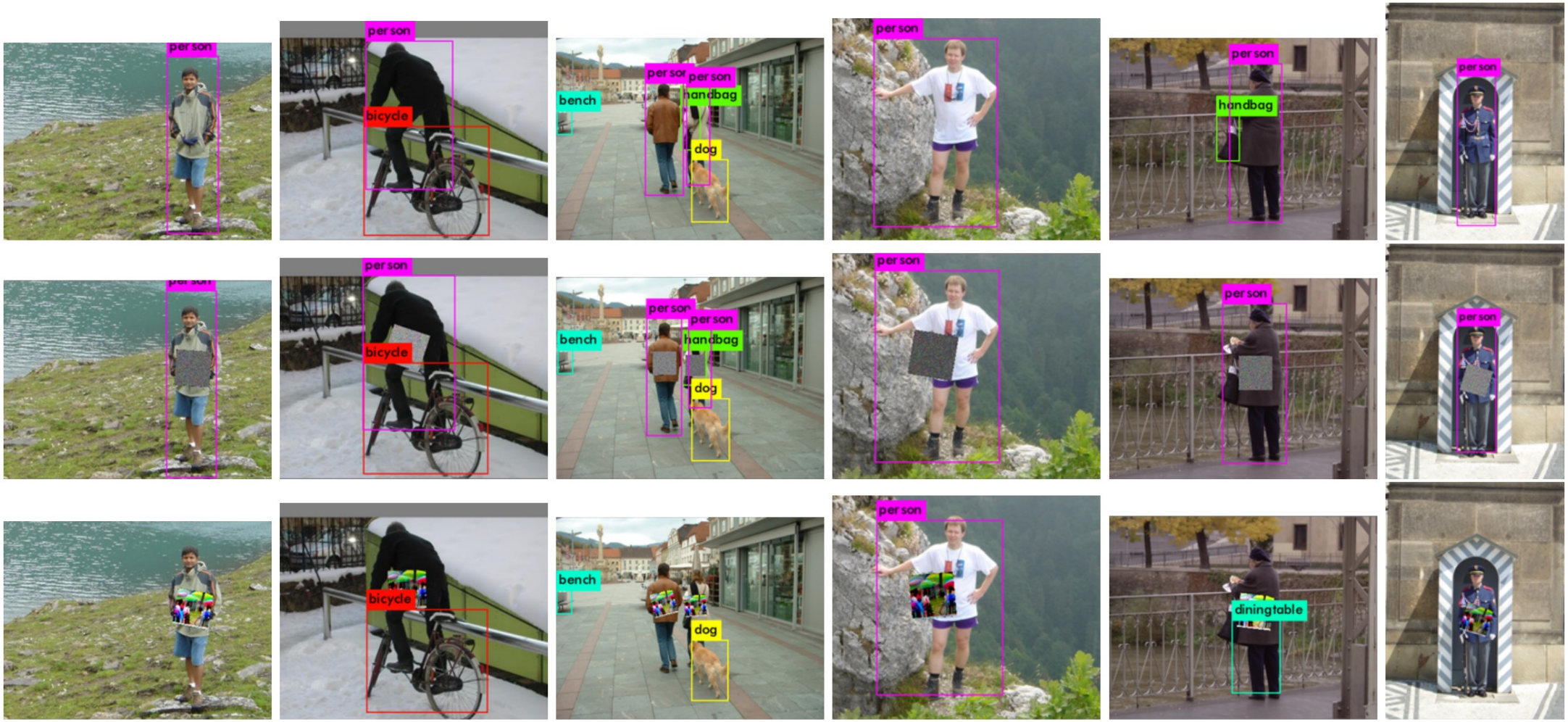


Figure 3: Overview of the pipeline to get the object loss.


person
bicycle
bench
handbag
dog
diningtable

Thank You

Questions?