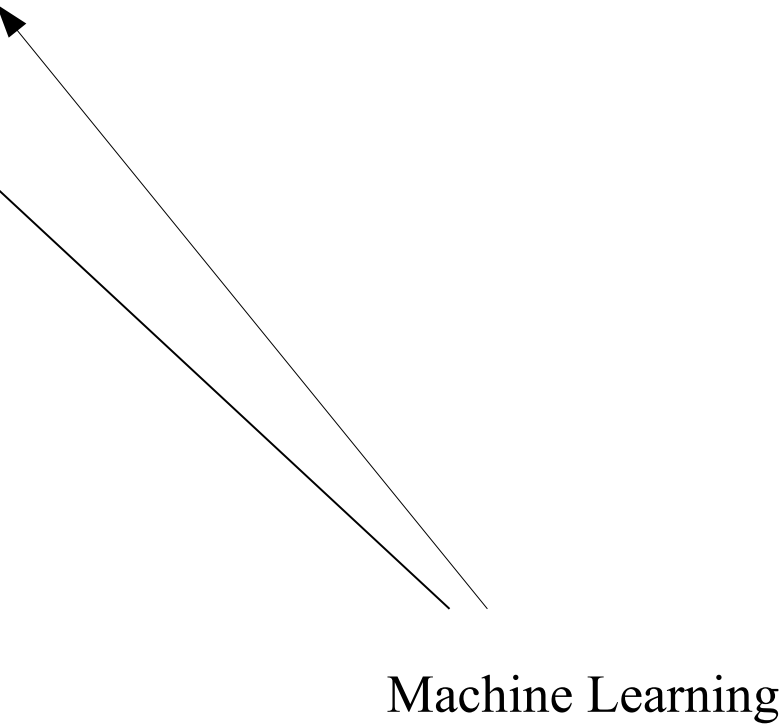
ICT Risk Assessment

Fabrizio Baiardi
f.baiardi@unipi.it

Syllabus

- New Technology
  - New Attacks
  - Countermeasures



Adversarial Machine Learning

- Several machine learning models, including neural networks, consistently misclassify adversarial examples
- These are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake; they're like optical illusions for machines
- Adversarial examples work across different mediums and securing systems against them can be difficult
- Adversarial examples are a good aspect of security to work on because they represent a concrete problem in AI security and safety that can be addressed in the short term, and because fixing them is difficult enough that it requires a serious research effort

An example of an adversarial example

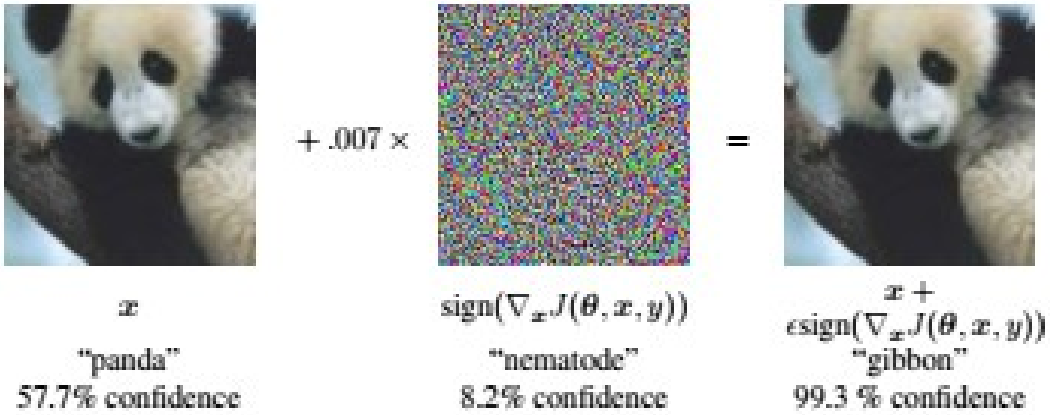


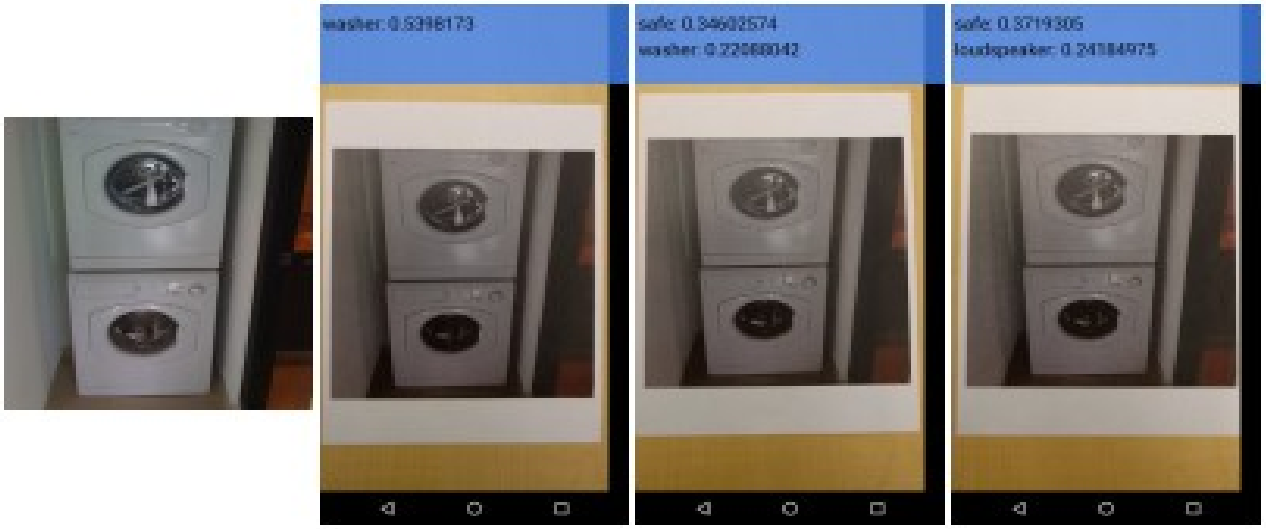
Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy
Google Inc., Mountain View, CA

Adversarial Machine Learning – Another def

- An adversarial example is a sample of input data which has been modified very slightly in a way that is intended to cause a machine learning classifier to misclassify it.
- In many cases, these modifications can be so subtle that a human observer does not even notice the modification at all, yet the classifier still makes a mistake.
- Adversarial examples can be used to attack machine learning systems, even if the adversary has no access to the underlying model.
- In the current a threat model in which the adversary can feed data directly into the machine learning classifier.
- This is can happen for systems operating in the physical world, for example those which are using signals from cameras and other sensors as an input.

Adversarial Machine Learning – Another def



(a) Image from dataset (b) Clean image (c) Adv. image, $\epsilon = 4$ (d) Adv. image, $\epsilon = 8$

Figure 1: Demonstration of a black box attack (in which the attack is constructed without access to the model) on a phone app for image classification using physical adversarial examples. We took a clean image from the dataset (a) and used it to generate adversarial images with various sizes of adversarial perturbation ϵ. Then we printed clean and adversarial images and used the TensorFlow Camera Demo app to classify them. A clean image (b) is recognized correctly as a "washer" when perceived through the camera, while adversarial images (c) and (d) are misclassified. See video of full demo at `https://youtu.be/zQ_uMenoBCk`.

Adversarial Examples for Linear Model

- In many problems, the precision of an individual input feature is limited. For example, digital images often use only 8 bits per pixel so they discard all information below 1/255 of the dynamic range.
- Because the precision of the features is limited, it is not rational for the classifier to respond differently to an input x than to an adversarial input $x' = x + \eta$ if every element of the perturbation η is smaller than the precision of the features.
- Formally, for problems with well-separated classes, we expect the classifier to assign the same class to x and x' so long as $\|\eta\|_\infty < \varepsilon$, where ε is small enough to be discarded by the sensor or data storage apparatus associated with our problem.

Adversarial Examples for Linear Model - I

- $w^t x' = w^t x + w^t \eta$ is the dot product between a weight vector w and an adversarial example x'
- The adversarial perturbation causes the activation to grow by $w^t \eta$. This increase can be maximised by assigning $\eta = \text{sign}(w)$.
- If w has n dimensions and the average magnitude of an element of the weight vector is m, then the activation will grow by mn.
- Since $\|\eta\|_\infty$ does not grow with the dimensionality of the problem but the change in activation caused by perturbation by η can grow linearly with n, then for high dimensional problems, we can make many infinitesimal changes to the input that add up to one large change to the output.
- This as a sort of "accidental steganography," where a linear model is forced to attend exclusively to the signal that aligns most closely with its weights, hence a simple linear model can have adversarial examples if its input has sufficient dimensionality

Adversarial Examples for Linear Model - II

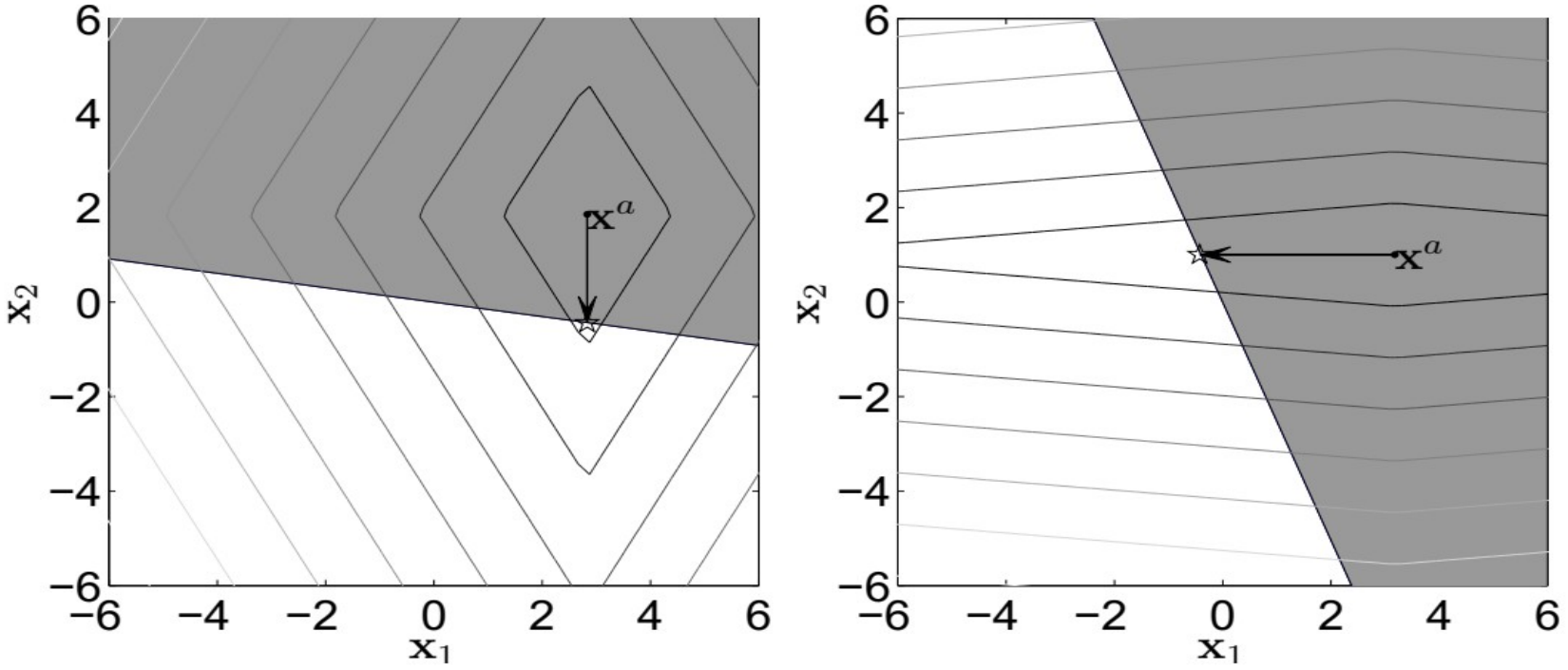
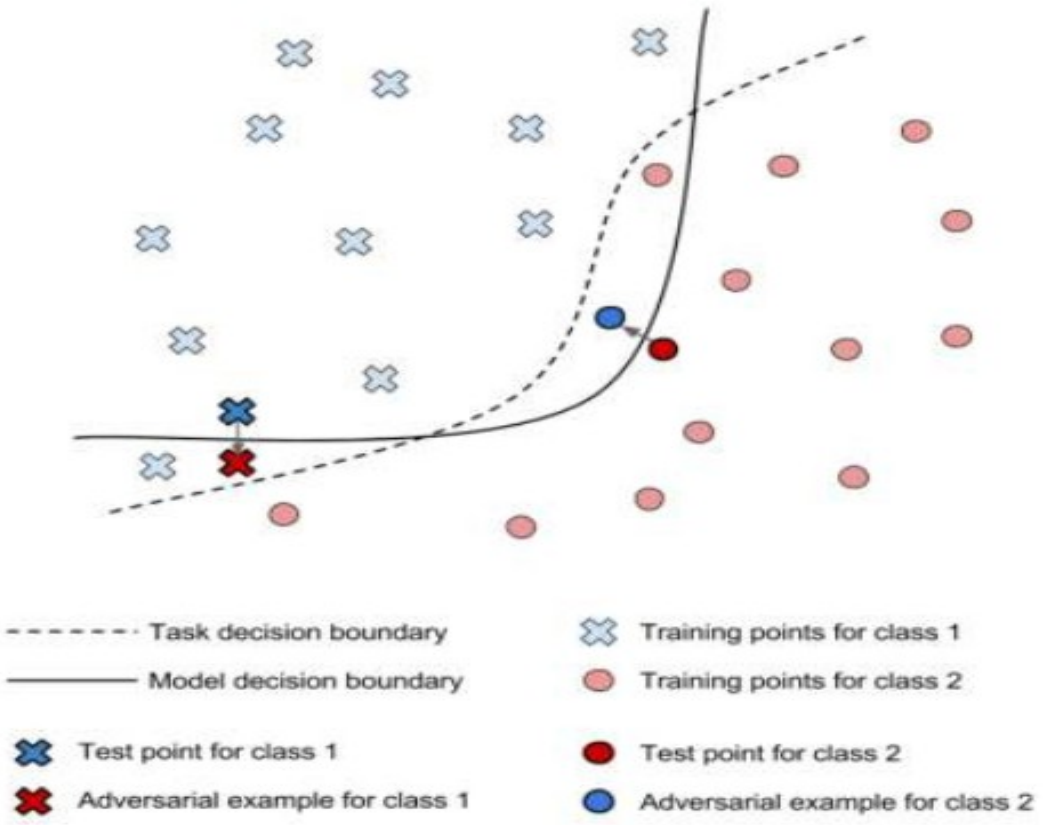


Fig. 2. Searching $\mathbf{x}^*$ (denoted by a star) in a 2-dim. space with a linear decision boundary. The shaded area demonstrates the positive response of the classifier. Arrow represents the optimal searching direction. **(left)** The optimal searching direction is along x_2. **(right)** The optimal searching direction is along x_1.

Adversarial Examples for Linear Model - III



Adversarial Examples for Linear Model - III

- Many threat models assume the adversary knows the machine learning parameters to solve the optimization problem for the input perturbation.
- In more realistic models, an adversary can only observe the model predictions on chosen inputs. As an example, it can figure out
  - how to design webpages that are well ranked
  - how to craft spam that evades detection.
- In these black-box settings, the machine learning model is said to act as an oracle. An attack strategy
  - queries the oracle to approximate its decision boundaries
  - usea this model to craft examples the oracle misclassifies
- The attacks exploit the transferability of adversarial examples: they are often misclassified simultaneously across different models solving the same machine learning task, despite the fact that these models differ in their architecture or training data

Adversarial Examples of Deep Networks

- The criticism of deep networks as vulnerable to adversarial examples is somewhat misguided because they can *represent* functions that resist adversarial perturbation.
- The universal approximator theorem guarantees that a neural network with at least one hidden layer can represent any function to an arbitary degree of accuracy so long as its hidden layer is permitted to have enough units.
- Shallow linear models are not able to become constant near training points while also assigning different outputs to different training points.
- The universal approximator theorem does not say anything about whether a training algorithm will be able to discover a function with all of the desired properties.
- Obviously, standard supervised training does not specify that the chosen function be resistant to adversarial examples. This must be encoded in the training procedure somehow.

Defenses against Adversarial Examples

- Traditional techniques for making machine learning models more robust, such as weight decay and dropout, generally do not provide a practical defense against adversarial examples. So far, only two methods have provided a significant defense
  - Adversarial training
  - Defensive distillation

- Simple defense strategies such as "gradient masking" do not work.

- Even these specialized algorithms can easily be broken by giving more computational firepower to the attacker.

Gradient Masking

- Shattered gradients are nonexistent or incorrect gradients caused either intentionally through nondifferentiable operations or unintentionally through numerical instability.
- Stochastic gradients are gradients that depend on testtime entropy unavailable to the attacker.
- Vanishing/exploding gradients in very deep or recurrent computation that consist of multiple iterations of neural network evaluation, feeding the output of one computation as the input of the next. This type of computation, when unrolled, can be viewed as an extremely deep neural network evaluation, which can cause vanishing/exploding gradients.

Adversarial Training

- Adversarial training seeks to improve the generalization of a model when presented with adversarial examples at test time by proactively generating adversarial examples as part of training
- This idea is not new but was not yet practical because of the high computation cost of generating adversarial examples.
- Methods exist (fast gradient sign method) to generate large batches of adversarial examples during the training process
- The model is then trained to assign the same label to the adversarial example as to the original example
- We might take a picture of a cat, and adversarially perturb it to fool the model into thinking it is a vulture, then tell the model it should learn that this picture is still a cat.

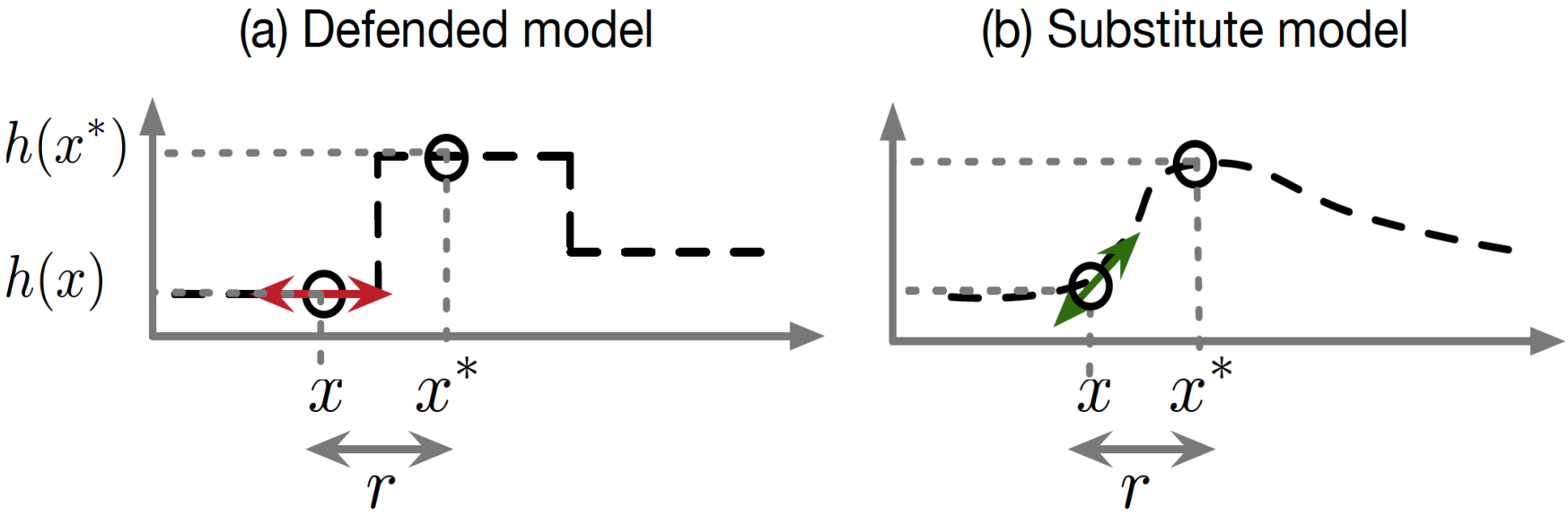
Defensive Distillation

- Defensive distillation smooths the model's decision surface in adversarial directions exploited by the adversary.
- It is a training procedure where one model is trained to predict the probabilities output by another model that was trained earlier.
- Its goal is that the final model's responses is more smooth, so it works even if both models are the same size.
- The reason it works is that the first model is trained with "hard" labels (100% probability that an image is a dog rather than a cat) and then provides "soft" labels (95% probability that an image is a dog rather than a cat) to train the second model.
- The second distilled model is more robust to attacks

Gradient Masking (why it does not work)

- The defense strategies that perform gradient masking typically result in a model that is very smooth in specific directions and neighborhoods of training points, which makes it harder for the adversary to find gradients indicating good candidate directions to perturb the input in a damaging way for the model.
- The adversary can train a substitute model: a copy that imitates the defended model by observing the labels that the defended model assigns to inputs chosen carefully by the adversary.
- The adversary can then use the substitute model's gradients to find adversarial examples that are misclassified by the defended model as well.
- The gradient masking phenomenon would be exacerbated for higher dimensionality problems

Gradient Masking (does not work)



Defensive Strategies - I

- Gradient masking is not a very good defense because it defends against an attacker who uses the gradient, but if the attacker knows we are using that defense, it can just switch to a transferability attack. This means that gradient masking is not an adaptive defense.

- Most defenses against adversarial examples that have been proposed so far just do not work very well at all, but the ones that do work are not adaptive. This means it is like they are playing a game of whack-a-mole: they close some vulnerabilities, but leave others open.

Defensive Strategies – II

- Adversarial examples are hard to defend against because it is hard to construct a model of the adversarial example crafting process.
- Adversarial examples solve an optimization problem that is non-linear and non-convex for many ML models. Because we don't have good tools for describing the solutions to these problems, it is very hard to prove that a defense will rule out a set of examples.
- From another point of view, adversarial examples are hard to defend against because ML models should produce good outputs for every possible input. Most ML models work very well but only on a very small amount of all the possible inputs they might encounter.
- Because of the massive amount of possible inputs, it is very hard to design a defense that is truly adaptive.

Other attacks

- Beside test-time inputs to confuse a ML model, other kinds of attacks are possible, such as those that surreptitiously modifies the training data so that the model learns to behave the way the attacker wishes.
- The no free lunch theorem for supervised learning says that, averaged over all possible datasets, no ML algorithm does better on new points at test time than any other algorithm.
- At first glance, this seems to suggest that all algorithms are equally vulnerable to adversarial examples.
- However, the theorem does not assume anything about the structure of the problem. Adversarial examples assume that small perturbations of the input should not change the output class, so the theorem in its typical form does not apply.

Open problems

- The study of adversarial examples is exciting because many of the most important problems remain open, both in terms of theory and in terms of applications.

- On the theoretical side, no one yet knows whether defending against adversarial examples is a theoretically hopeless endeavor (like trying to find a universal machine learning algorithm) or if an optimal strategy would give the defender the upper ground (like in cryptography).

- On the applied side, no one has yet designed a truly powerful defense algorithm that can resist a wide variety of adversarial example attack algorithms.

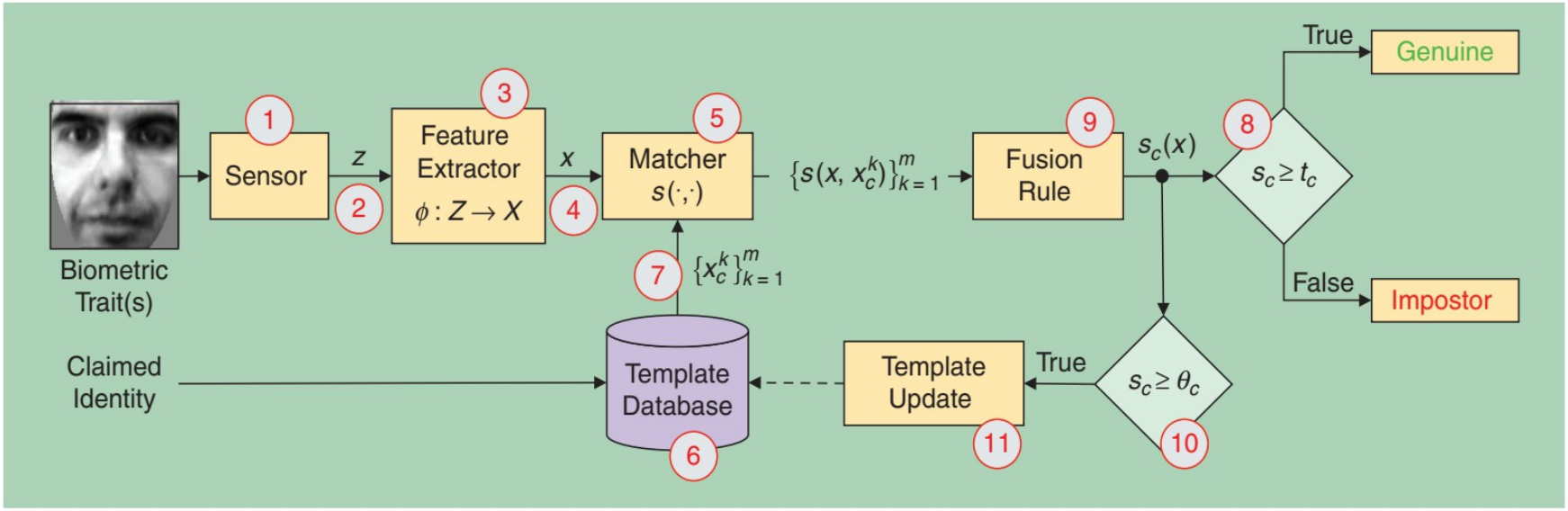
The challenge of verification and testing

- The limitations of existing defenses point to the lack of verification of machine learning models.
- Testing = evaluating the system in several conditions and observing its behavior, watching for defects.
- Verification = producing a compelling argument that the system will not misbehave under a very broad range of circumstances.
- Orthogonal to this issue is the question of which input values should be subject to verification and testing. Do we intend to verify or test the system only for "naturally occurring" legitimate inputs, or do we intend to provide guarantees for its behavior on arbitrary, degenerate inputs?
- ML has traditionally relied primarily on testing. A classifier is usually applied to several examples from a test set and measuring its accuracy on these examples. By definition, these testing procedures cannot find all of the possible—previously unseen—examples that may be misclassified.

Verification, testing and robustness

- Besides using verification rather than testing, ensure that the model will behave safely on inputs crafted by an attacker = guarantee robustness to adversarial examples.
- The natural way to test robustness to adversarial examples is to evaluate the model accuracy on a test set adversarially perturbed to create adversarial examples. This applies traditional ML testing methodology to new inputs.
- Testing cannot provide security guarantees, because ***the attacker inputs can differ from those used for testing***. For example, a model that is tested and found to be robust against some methods of adversarial example generation may be vulnerable to more computationally expensive methods.
- Testing only provides a lower bound on the failure rate eg if testing identifies n inputs that cause failure it concludes that **at least n inputs cause failure**
- To provide security guarantees, an upper bound is necessary eg we need a means of becoming reasonably confident that **at most n inputs cause failure**.

An example. Adversarial attacks on biometrics



[FIG1] The architecture of a biometric verification system and corresponding attack points, highlighted with red-circled numbers. During verification, the image $z \in \mathcal{Z}$ (e.g., a face image) acquired by the sensor is processed by a feature extractor $\phi: \mathcal{Z} \mapsto \mathcal{X}$ to obtain a compact representation $x \in \mathcal{X}$ (e.g., a graph). The templates $\{x_c^k\}_{k=1}^m$ of the claimed identity c are retrieved from the template database, and compared to x using a matching algorithm $s: \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. The resulting scores $\{s(x, x_c^k)\}_{k=1}^m$ are combined by a fusion rule, producing an aggregated score $s_c(x)$ that expresses the degree to which x is likely to belong to c. The score $s_c(x)$ is then compared with a decision threshold t_c to decide whether the claim is genuine or impostor. If a template self-update is implemented, and $s_c(x)$ is not lower than a self-update threshold θ_c, one of the templates in $\{x_c^k\}_{k=1}^m$ is updated depending on x, according to a given policy.

An example. Adversarial attacks on biometrics

ATTACK TECHNIQUE	ATTACK LOCATION	ATTACK POINT(S)	DEFENSE
SPOOFING	SENSOR	1	LIVENESS DETECTION, MULTIBIOMETRICS (SECURE FUSION)
REPLAY	INTERFACES/CHANNELS	2, 4, 7	ENCRYPTED CHANNEL, TIME STAMP, CHALLENGE-RESPONSE, PHYSICAL ISOLATION
HILL CLIMBING	INTERFACES/CHANNELS	2, 4	ENCRYPTED CHANNEL, TIME STAMP, CHALLENGE-RESPONSE, PHYSICAL ISOLATION, SCORE QUANTIZATION
MALWARE INFECTION	MODULES/ALGORITHMS	3, 5, 8–11	SECURE CODE, SPECIALIZED HARDWARE, ALGORITHMIC INTEGRITY
TEMPLATE THEFT, SUBSTITUTION, AND DELETION	TEMPLATE DATABASE	6	TEMPLATE ENCRYPTION, CANCELABLE/REVOKABLE TEMPLATES

Poisoning

- The attacker's capability consists of modifying the template database, either by directly manipulating it (e.g., through malware infection), or, more realistically, by submitting fake traits that are erroneously used to update the template gallery of a given client.
- In terms of security violation, an integrity violation thus amounts to replacing a victim's template with an attacker's template or to adding an attacker's template in the victim's gallery.
- This allows the attacker to impersonate the victim without using any further spoofing or replay attack, but directly using her own biometric trait.
- The goal of an availability violation is to cause a denial of service, instead, by replacing or compromising the majority of templates in the victim gallery. This will indeed deny the victim access to the system.

Direct Poisoning

- Direct poisoning is implemented by inserting points in a training dataset used for anomaly detection
- This can gradually shift the decision boundary of a simple centroid model, i.e. a model that classifies a test input as malicious when it is too far from the empirical mean of the training data.
- The detection model is learned in an online fashion
  - new training data is collected at regular intervals
  - the parameter values are computed based on a sliding window
- Injection of poisoning data in the training dataset is a particularly easy task for adversaries in these online settings.
- Poisoning points are found by solving a linear programming problem that maximizes the displacement of the centroid (empirical mean of the training data).
- This is made possible by the simplicity of the centroid model

Indirect Poisoning

- Adversaries poison the model's training data before its pre-processing
- For instance, perturbation are inserted into worm traffic flows to prevent Polygraph, a worm signature generation tool, from learning meaningful signatures .
- Polygraph combines a flow tokenizer together with a classifier that determines whether a flow should be in the signature.
- Polymorphic worms are crafted with noisy traffic flows such that
  (1) their tokenized representations will share tokens not representative of the worm's traffic flow,
  (2) they modify the classifier's threshold for using a signature to flag worms.
- This attack forces Polygraph to generate signatures with tokens that do not correspond to invariants of the worm's behavior.
- This has also been applied to to feature selection algorithms like LASSO