QuantifiCare

A.I. seminar

Challenges of A.I. : integrity, confidentiality and availability

QuantifiCare

Adversarial Examples

Summary

Reminders, formalism and notations

Reminders, formalism and notations

The l_p norm of a vector $u \in \mathbb{R}^d$ for $0 < p < \infty$, denoted by $\|u\|_p$, corresponds to:

$$\|u\|_p = \left(\sum_{i=1}^{d} |u_i|^p \right)^{\frac{1}{p}}$$

The l_∞ norm of a vector $u \in \mathbb{R}^d$, denoted by $\|u\|_\infty$, corresponds to:

$$\|u\|_\infty = \max_{i=1}^{d} |u_i|$$

The l_0 norm of a vector $u \in \mathbb{R}^d$, denoted by $\|u\|_0$, corresponds to:

$$\|u\|_0 = \sum_{i=1}^{d} \mathbb{1}_{\{u_i \neq 0\}}$$

Reminders, formalism and notations

The l_p ball of center c and radius r, corresponding to $\left\{u \mid \|u - c\|_p \leq r\right\}$, is denoted by $B_p(c, r)$. $B_p(r)$ simply denotes the l_p ball of radius r centered at origin.

We denote by $\text{proj}_{B_p(c,r)}(u)$ the projection of a vector u onto the l_p ball of center c and radius r.

Considering a function f, such that:

$$f : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_3}$$
$$x_1 \times x_2 \mapsto y$$

The gradient of the function f at point (x_1, x_2) with respect to the variable x_1 is denoted as $\nabla_{x_1} f(x_1, x_2)$.

Reminders, formalism and notations

Considering two discrete probability distributions p and q with support A, the cross entropy is given by:

$$CE(p, q) = -\sum_{a \in A} p(a) \log q(a)$$

The Kullback-Leibler divergence between p and q is given by:

$$KL(p \| q) = \sum_{a \in A} p(a) \frac{\log p(a)}{\log q(a)}$$

$\mathcal{N}(m, \sigma^2)$ denotes the Gaussian distribution with mean m and standard deviation σ.

Reminders, formalism and notations

QuantifiCare

Classification task for a neural network

Goal:

Map each input from an input space $\mathcal{X} \subset \mathbb{R}^d$ to one of the K possible class labels from the label space $\mathcal{Y} = \{1, 2, \ldots, K\}$.

We suppose that input-label pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are sampled independently from an unknown probability distribution $\mathcal{D}$.

Reminders, formalism and notations

QuantifiCare

Classification task for a neural network

We consider a neural network model M_θ (noted M except when necessary).

Objective: Assign to an input $x \in \mathcal{X}$ a confidence score to each class, in the form of a *confidence score vector* $F(x) = (F_1(x), F_2(x), \ldots, F_K(x)) \in [0,1]^K$, where $\sum_{c=1}^{K} F_c(x) = 1$.

Predicted label: $M(x) = \text{argmax}_i \, F_i(x)$.

Score function of M: $F : \mathcal{X} \mapsto [0,1]^K$

Logits: output vector at the final layer, without considering an activation function, denoted by $h(x) = (h_1(x), h_2(x), \ldots, h_K(x))$.

Reminders, formalism and notations

QuantifiCare

Classification task for a neural network

The *confidence score vector* is obtained by applying the softmax function to the logits:

$$F_i(x) = \frac{e^{h_i(x)}}{\sum_{j=1}^{K} e^{h_j(x)}}$$

Output vector at the penultimate layer of a neural network model: $g(x)$.
(corresponds to what is often called *features* in the deep learning litterature).

The *logits* are a linear combination of the values in $g(x)$, and so the prediction of the model can be seen as learning a linear classifier on these feature values.

Reminders, formalism and notations

QuantifiCare

Classification task for a neural network

Pipeline for the prediction scheme for an input x:

$$x \in \mathcal{X} \longrightarrow \cdots \longrightarrow g(x) \longrightarrow h(x) \in \mathbb{R}^K \xrightarrow{softmax} F(x) \in [0,1]^K \xrightarrow{argmax} M(x) \in \mathcal{Y}$$

Reminders, formalism and notations

QuantifiCare

Training process

To perform the training phase of the model M_θ, need to first define a cost function:

$$\mathcal{L}(x, y, M_\theta) : \mathcal{X} \times \mathcal{Y} \times [0, 1] \mapsto \mathbb{R}$$

indicates how much the predicted label given by M_θ fits the ground-truth label y (the smaller the more accurate the model is).

Objective:

Obtain a model minimizing the cost function for input-label pairs (x, y) drawn from $\mathcal{D}$:

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(x, y, M_\theta)\right]$$

Reminders, formalism and notations

QuantifiCare

Training process

Distribution $\mathcal{D}$ is unknown
$\Rightarrow$ training phase of M is performed in practice by minimizing the *empirical error*, which resumes to minimizing the cost function on the available training set $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$.

Practically, the training phase of M_θ therefore resumes to the *empirical risk minimization* problem, given as:

$$\min_\theta \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(x_i, y_i, M_\theta)$$

Reminders, formalism and notations

QuantifiCare

Training process

In practice, solving directly the empirical risk minimization problem is intractable, then the Stochastic Gradient Descent (SGD) method is performed.

SGD: updating iteratively the parameters θ, by considering – at each step – a batch of samples of the train set, instead of the whole train set.

At each iteration, a batch of $B < n$ input-label pairs is drawn, $(x_1, y_1), (x_2, y_2), \ldots (x_B, y_B)$, and the parameters are updated as follows:

$$\theta \leftarrow \theta - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(x_i, y_i, M_\theta)$$

θ: learning rate

Reminders, formalism and notations

QuantifiCare

Training process

For a classification task, a typical example of a cost function is the cross-entropy loss.

Considering an input-label pair (x, y), the cross-entropy loss measures the adequacy between the model confidence score vector $F(x)$ and the one-hot encoded vector version of y, denoted as y_{hot}.

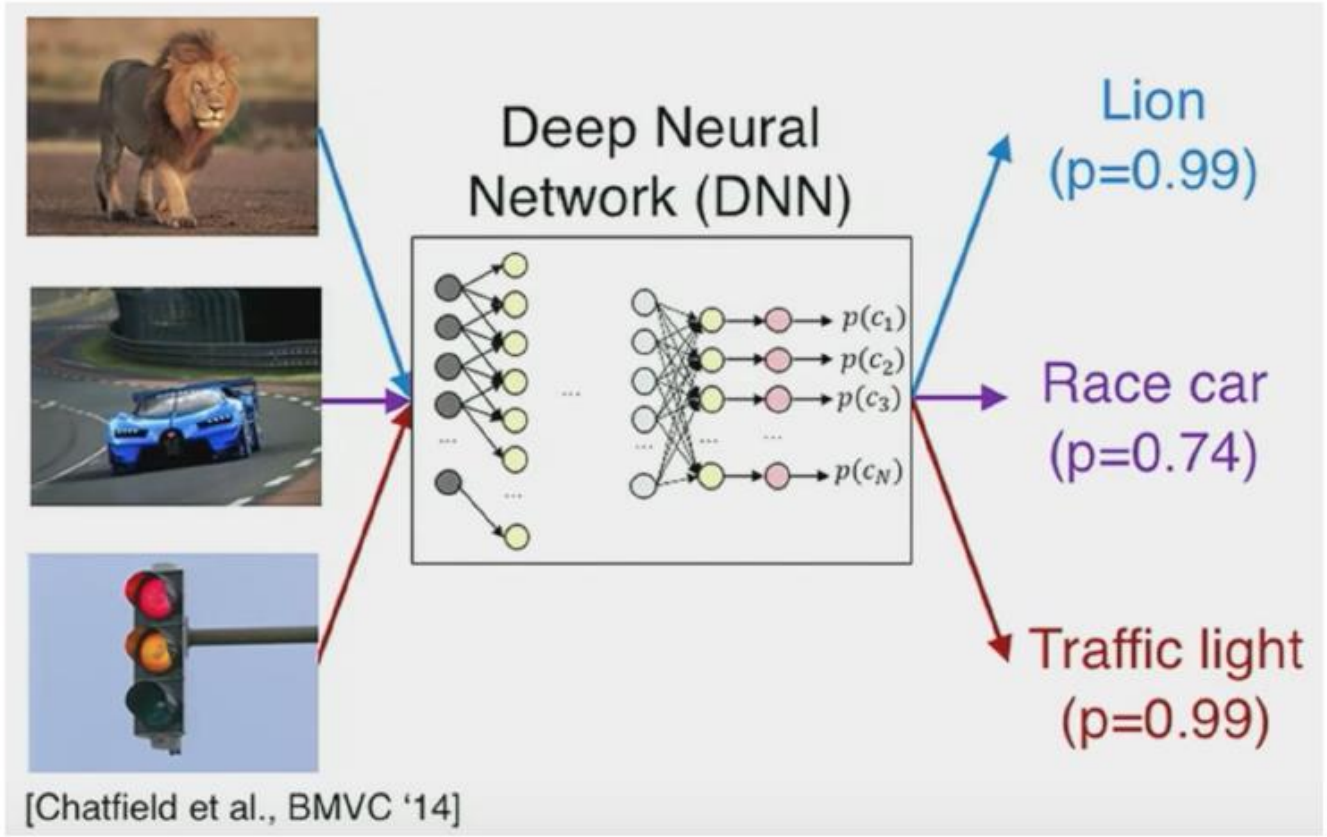
$$\mathcal{L}_{CE}(x, y, M_\theta) = CE(y_{hot}, F(x)) = -\sum_{c=1}^{K} \mathbb{1}_{\{c=y\}} \ \log \ F_c(x)$$

The phenomenon of adversarial examples

The phenomenon of adversarial examples

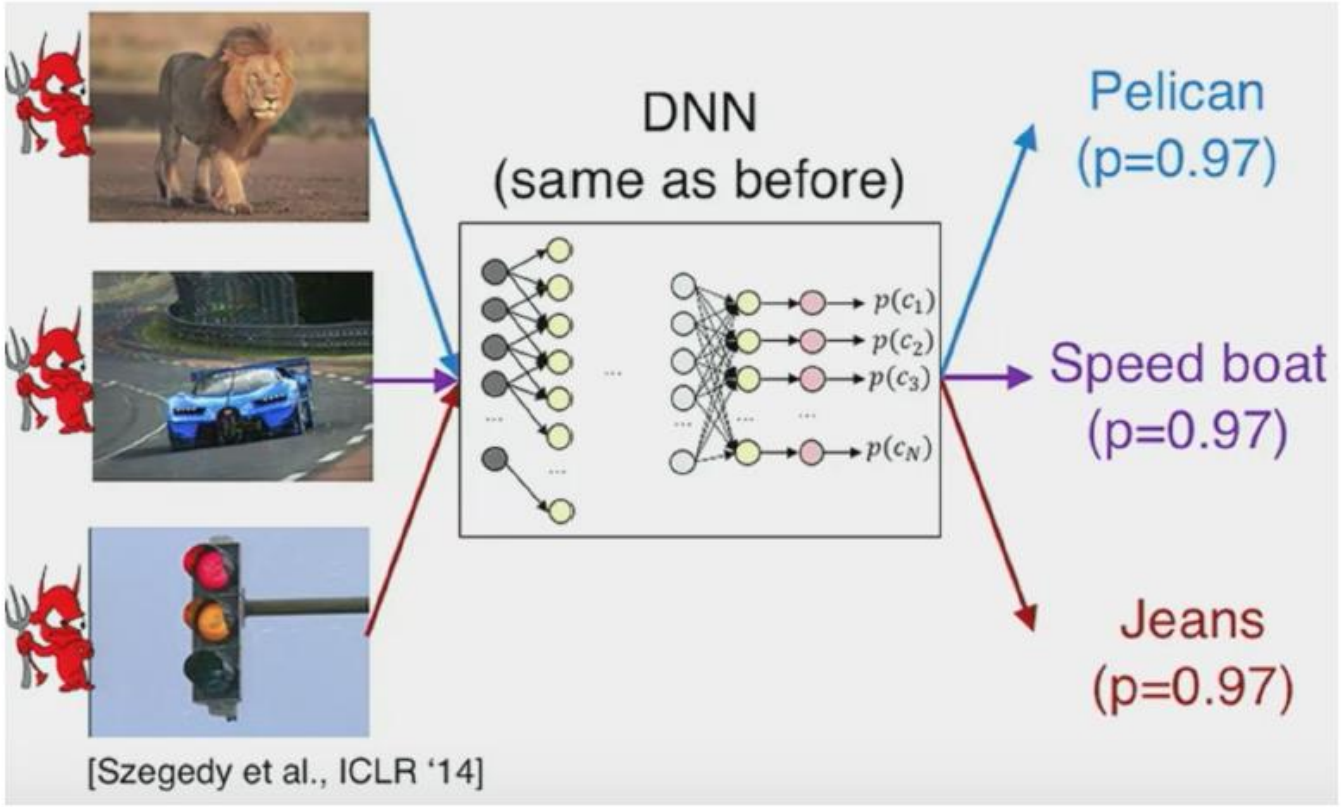
QuantifiCare

Reminder:



The phenomenon of adversarial examples

Reminder:



The phenomenon of adversarial examples

QuantifiCare

Reminder:

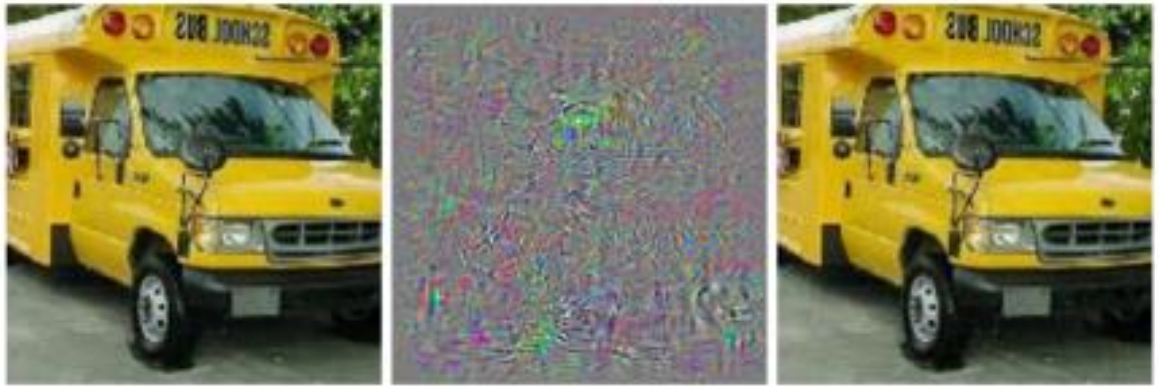
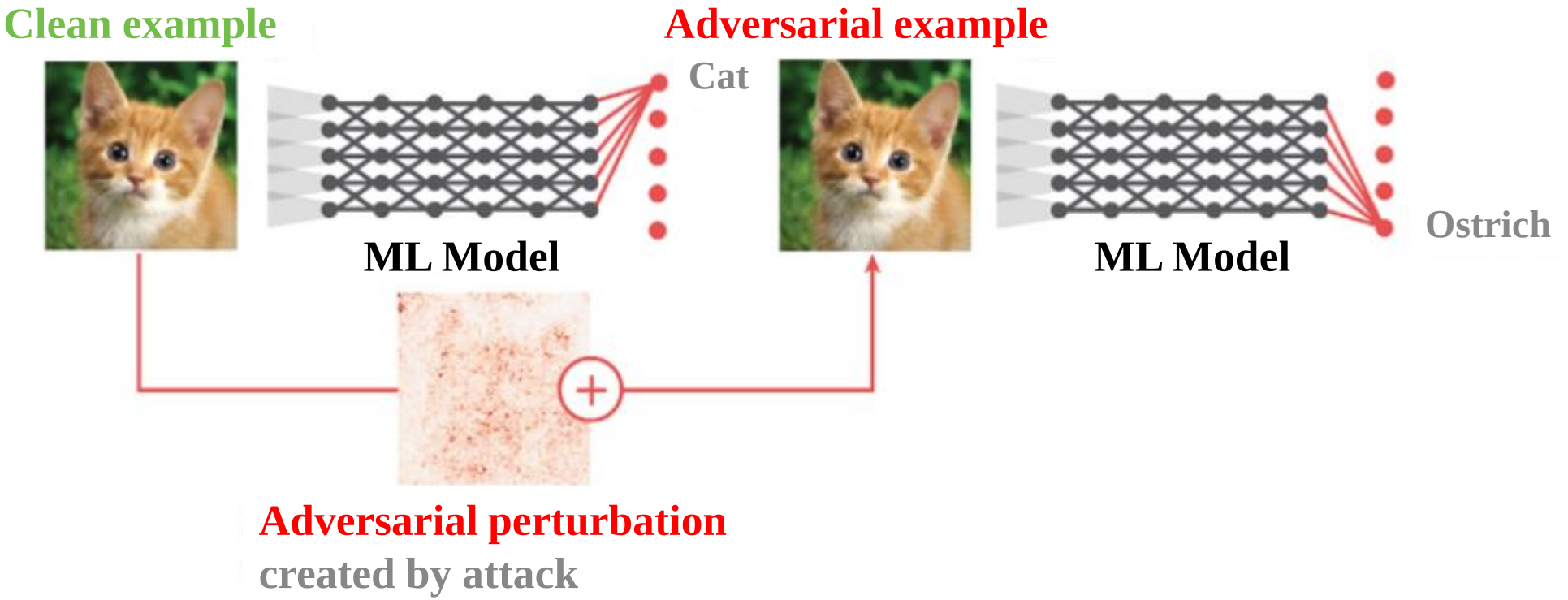


Figure 2.9: Illustration of adversarial examples from [11]. Clean image classified as a "school bus"(left), magnified adversarial perturbation (middle), and resulting adversarial example misclassified by the target model as "ostrich" (right).

The phenomenon of adversarial examples

Adversarial example = Clean example + adversarial perturbation



The phenomenon of adversarial examples

Brief history of adversarial examples

« Intriguing properties of neural networks », 2013, Szegedy *et al.*, ICLR 2014

"counter-intuitive properties" of neural network, misclassifying examples, which are "indistinguishable from the regular examples"

« Evasion attacks against machine learning at test time », 2013, Biggio *et al.*, ECMLKDD 2013

"evasion attacks": examples modified in order to fool a classifier at inference time. The notion of "indistinguishability" of the perturbation is not at the core of this work

« Explaining and harnessing adversarial examples», 2013, Goodfellow *et al.*, ICLR 2015

Introduce a method to craft adversarial examples

The phenomenon of adversarial examples

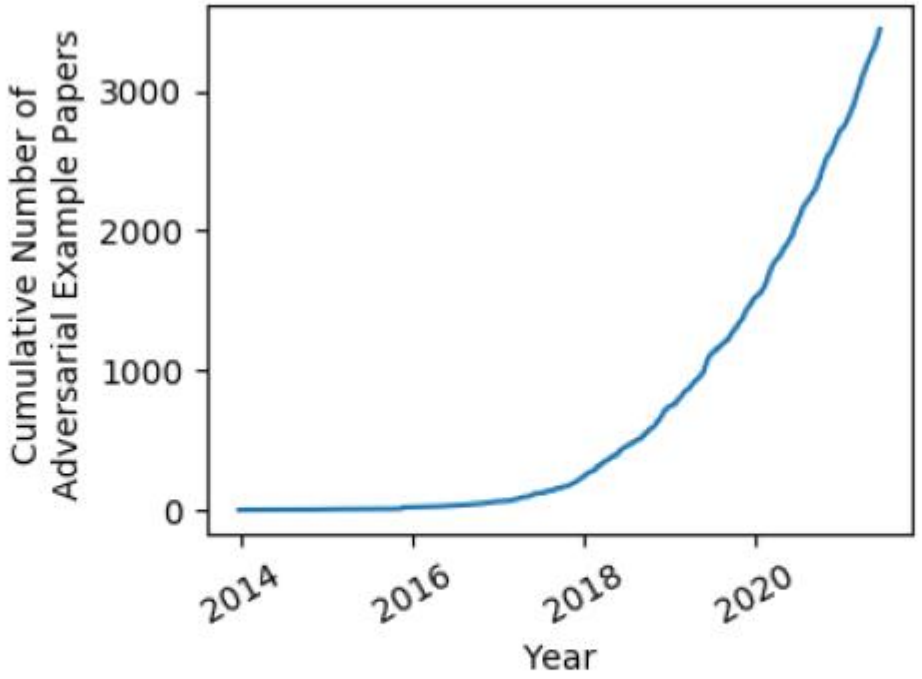


Figure 4.1: From Nicholas Carlini [74]. Number of adversarial examples related papers over the years (extracted from Arxiv).

The phenomenon of adversarial examples

Brief history of adversarial examples

In big conferences, a lot a papers submitted regarding defense methods against adversarial examples but these methods were quickly defeated after

=> incentive nowadays to properly define the threat model, and meticulously evaluate the robustness with adaptive adversaries

\+ Research does not limit to the l_p norm for the distance between clean and adversarial examples, as introduced in the seminal work of Szegedy et al. and Goodfellow et al.,
=> other types of distances such as the Wasserstein distance and perturbations such as rotations and translation and unrestricted but localized perturbations

The phenomenon of adversarial examples

Digital and real-world adversarial examples

Important distinction: the level at which an adversarial examples is considered in a system involving neural networks.

Digital (pixel) world => perturbations considered directly at pixel level

(e.g. classification API)

Real (physical) world => perturbations implemented on real-world objects and then go through sensors

(images are captured with sensors and are then digitalized)

Real-world noise, climate conditions, …

The phenomenon of adversarial examples

Digital and real-world adversarial examples

Examples:

Kurakin et al. [100] show that adversarial examples printed on a sheet and then fed to a neural network classifier through a cell phone camera can fool this classifier.

Eykholt et al. [59] perform real-world adversarial attacks in the form of black and white stickers added to real-world traffic road signs, which fool a neural network classifier recording images at varying distances and angles of view.

Sitawari et al. [101] also propose a method to make advertising road signs, reasonably rejected by the traffic road sign recognition pipeline, being classified as precise traffic road signs.

The phenomenon of adversarial examples

Digital and real-world adversarial examples

Figure 4.2: Stop sign with adversarial black and white stickers, recognized as a "speed limit 45" road sign, from [59].

Figure 4.3: Clean advertising sign (left), reasonably rejected by the traffic sign recognition system, and its adversarial counterpart (right), recognized as a "stop" road sign, from [101].

The phenomenon of adversarial examples

Digital and real-world adversarial examples

Examples:

Boloor et al. [102] manage to deceive an autonomous car driving system by drawing adversarial black lines on the road, which would be considered by a human as simple tread marks.

Sharif et al. [58] design adversarial pair of glasses, which when worn by an individual, allows him to impersonate another individual or not to be detected by the system.

Zhou et al. [103] design a method for an adversary to fool a face-recognition system, which consists in illuminating the adversary's face with infrared light.

Thys et al. [104] design an adversarial image, which allows an individual not to be recognized as a person.

The phenomenon of adversarial examples

Digital and real-world adversarial examples

Figure 4.4: Physical-world infrared adversarial device from [103].

The phenomenon of adversarial examples

Digital and real-world adversarial examples

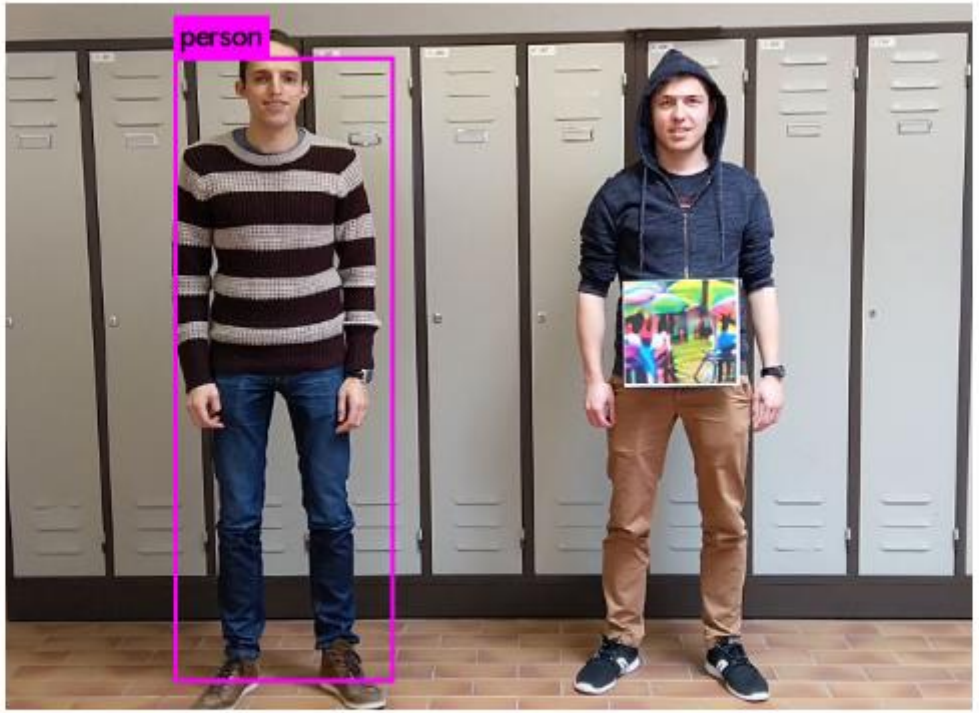


Figure 4.5: Physical attack against an individual detection system, from [104].

Threat model for adversarial examples

Threat model for adversarial examples

QuantifiCare

Adversary goal:

Target neural network model M_θ, trained for a classification task. As previously presented, adversarial examples, resulting of the malicious addition of an adversarial perturbation to a clean example, are crafted with a twofold objective:

- An oracle $\mathcal{O}$, such as a human, classifies in the same class the clean example and its adversarial counterpart.
- The target model M classifies the adversarial example in another class than the one it classifies the clean example in.

Threat model for adversarial examples

QuantifiCare

Adversary goal:

Targeted attacks

$$\text{Find } \delta \quad \text{s.t.} \begin{cases} \mathcal{O}(x+\delta) = \mathcal{O}(x) \\ M(x+\delta) = y_t \end{cases}$$

Untargeted attacks

$$\text{Find } \delta \quad \text{s.t.} \begin{cases} \mathcal{O}(x+\delta) = \mathcal{O}(x) \\ M(x+\delta) \neq M(x) \end{cases}$$

Threat model for adversarial examples

QuantifiCare

Adversary capacity:

Need for a mathematical proxy $\mathcal{D}$ for imperceptibility and benignness

Minimizing proxy => decreased suspiciousness

$$x' = x + \delta$$

The adversary capacity will set an upper bound ϵ on $\mathcal{D}(\delta)$

Under this upper bound, the perturbation is considered not to raise suspicion.

$$\text{Find } \delta$$
$$\text{s.t.} \begin{cases} \mathcal{D}(\delta) \leq \epsilon \\ M(x+\delta) \neq M(x) \end{cases}$$

Threat model for adversarial examples

QuantifiCare

Adversary capacity:

$\mathcal{D}$ stays a high level concept

l_0, l_1, l_2, l_∞ => adversary capacity: upper bound on $\|\delta\|_p$

Some other works:

i) Wasserstein distance
ii) Maximum translations and rotation values
iii) Which regions can be colored

Threat model for adversarial examples

QuantifiCare

Adversary capacity:

Most common adversarial capacities:

l_0 : the adversary is limited in the number of pixels he can modify

l_∞ : the adversary is limited in the maximum change for each pixel

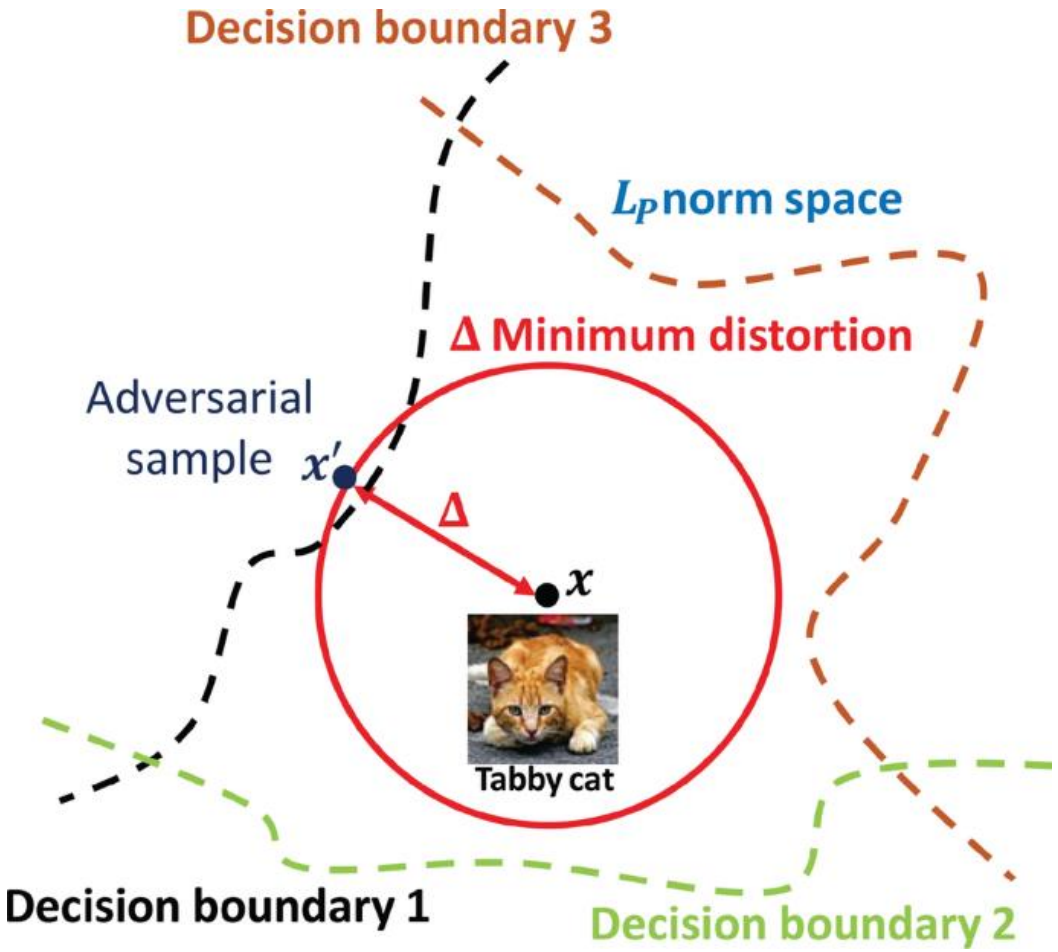
l_p : the adversary is limited in $\|x' - x\|_p$

Threat model for adversarial examples

QuantifiCare

Adversary capacity:

Example (l_2):



Threat model for adversarial examples

QuantifiCare

Adversarial knowledge:

1) White-box setting (worst case scenario)

Total access to the target model M (architecture, parameters, cost function, …)

Ability to compute gradients (gradient-based attacks)

(derivates of $\mathcal{L}, F, h, \ldots$)

=> Ability to quantify precisely the output varations to some input variation

\+ no limit on the number of queries to the target model

Threat model for adversarial examples

QuantifiCare

Adversarial knowledge:

2) Black-box setting

No ability to compute gradients

Different scenarios:

1) Only the output label => decision-based attacks

=> 2) Only the confidence score/logits => score-based attacks

\+ potential limit on the number of queries to the target model

Or …. Transferability (craft adversarial examples on a substitute model and transfer them to the target model). See later in presentation !

Threat model for adversarial examples

QuantifiCare

Adversarial knowledge:

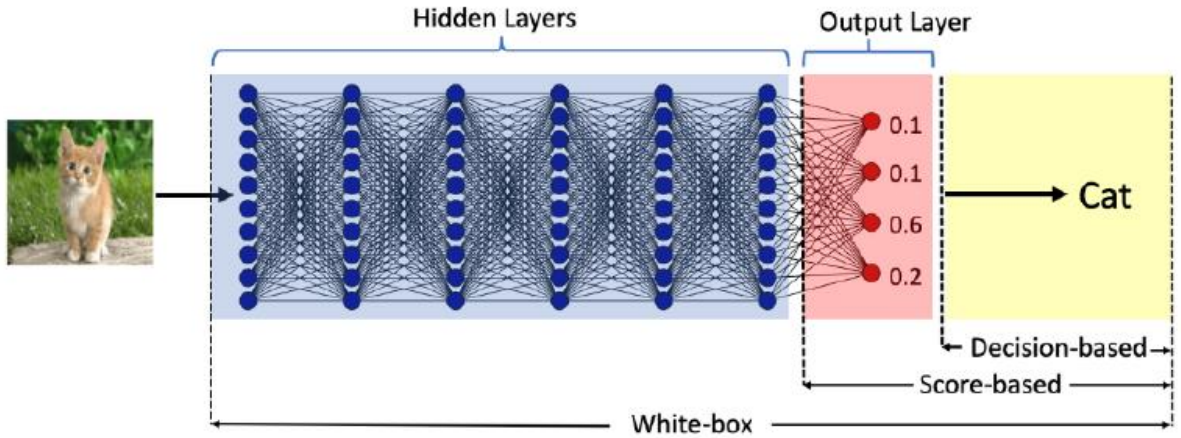


Figure 4.6: Illustration of the adversary capacity in the white and black-box settings, from [108]. In the black-box setting, the attacker may only access to the outputs of the model with different levels of precision (logits, confidence score vector, labels).

Threat model for adversarial examples

QuantifiCare

Adversarial knowledge:

Table 4.1: Knowledge of the adversary depending on the white-box or black-box setting. The sign $\sim$ means that the knowledge may vary depending on the tightness of the black-box setting.

Available knowledge of M_θ	*White-box*	*Black-box*
Gradients	✓	X
Predicted label $M(x)$	✓	✓
Confidence score vector $F(x)$	✓	$\sim$
Logits $h(x)$	✓	$\sim$
Architecture	✓	$\sim$
Data from $\mathcal{D}$	✓	$\sim$

Why do adversarial examples exist ?

Why do adversarial examples exist ?

Convolutional Neural Networks are biased towards texture

Geirhos et al. [113] show with experiments that Convolutional Neural Networks (CNN) tend to rely predominantly on local features such as textures rather than global shapes.

On the contrary, humans leverage to a lesser extent textures but rather use various global clues such as silhouettes and edges.

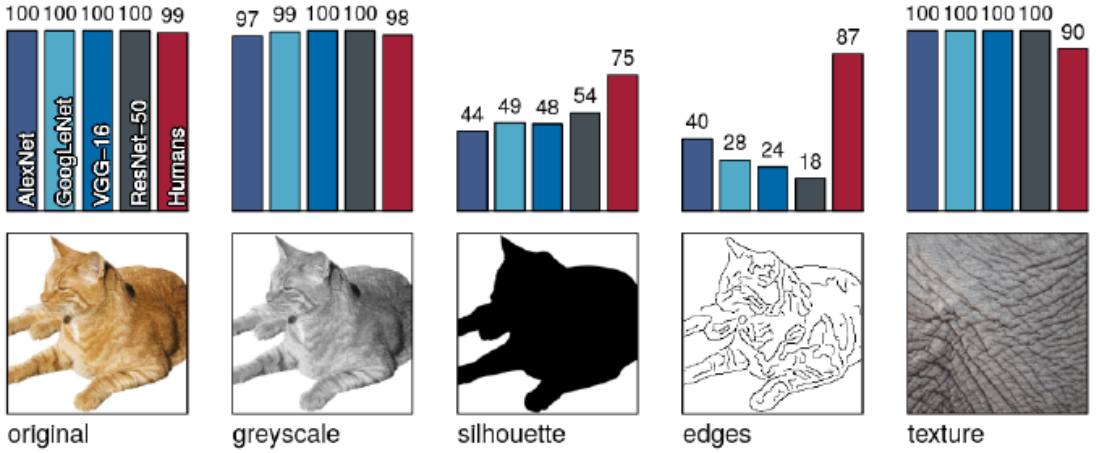


Figure 4.7: Accuracy of humans and models trained on Imagenet on standard and modified images. From left to right images, standard images, grayscale images, silhouette images, edge images and texture images, from [113]. Models seem to rely predominantly on texture to perform classification.

Why do adversarial examples exist ?

Robust Convolutional Neural Networks rely more on shape and contour

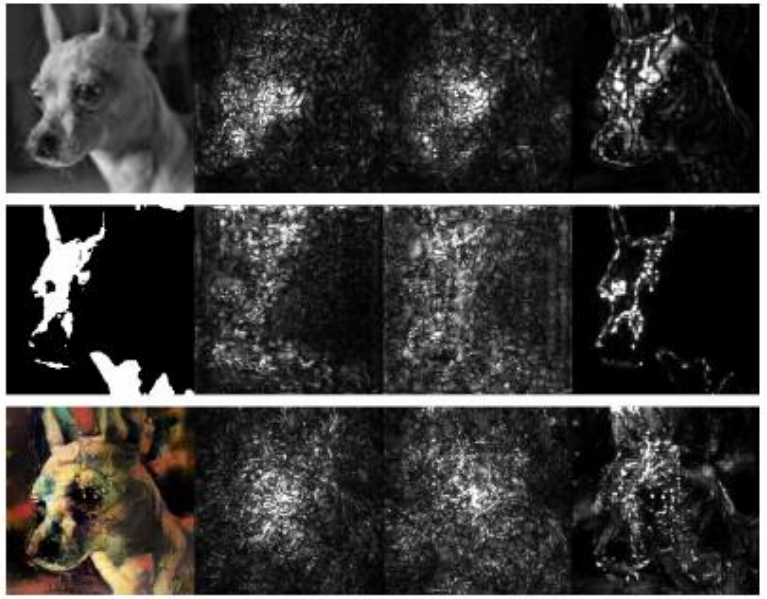
Zhang et al. [114] perform several experiments to analyze the concepts on which a CNN model trained with Adversarial Training (a defense scheme intended at making a model more robust against adversarial examples) relies to perform prediction. The three different concepts considered are **shape**, **contour** and **texture**.

- Stylizing: changing the style of an image (with style-transfer) => preserves the shape while destructing textures
- Saturation: preserves the contours while destructing the textures
- Patch-shuffling: splits the image into k x k parts and shuffles them, it preserves textures while destructing shape

Why do adversarial examples exist ?

Robust Convolutional Neural Networks rely more on shape and contour

A model showing a little accuracy decrease when evaluated on the stylized data set and a big accuracy decrease when evaluated on the patch-shuffled data set is very likely to rely on the shapes of the images, and not on local textures.



Sensitivity map: Sensitivity of the confidence score for the predicted class to variations of the input.

Figure 4.8: Sensitivity maps of different models on a standard image (top row), saturated image to keep contour (middle row) and stylized image to keep shape (bottom row). From left to right, standard model, underfitting model and model trained with Adversarial Training, from [114]. The sensitivity maps for standard models are more noisy than the ones for the model trained with Adversarial Training, more focused on shape and contours.

Why do adversarial examples exist ?

Fourier analysis of robustness

Yin et al. [90] suggest that the robustness of a classifier to some perturbations depends on the frequency of these perturbations and the frequency of the features the classifier relies on to perform prediction.

To support this hypothesis, four types of experiences are performed.

Why do adversarial examples exist ?

Fourier analysis of robustness

1) A standard model achieves 50% accuracy on data preprocessed with the use of a high-pass filter (invisible to the human eye).

2) Also also achieves > 30% accuracy on low-pass filtered images (indistinguishable colored shapes).

Ccl: Among all the statistics (color, texture, shape, etc.) for input/target relation, the classifier may have learned some which are not meaningful to humans.

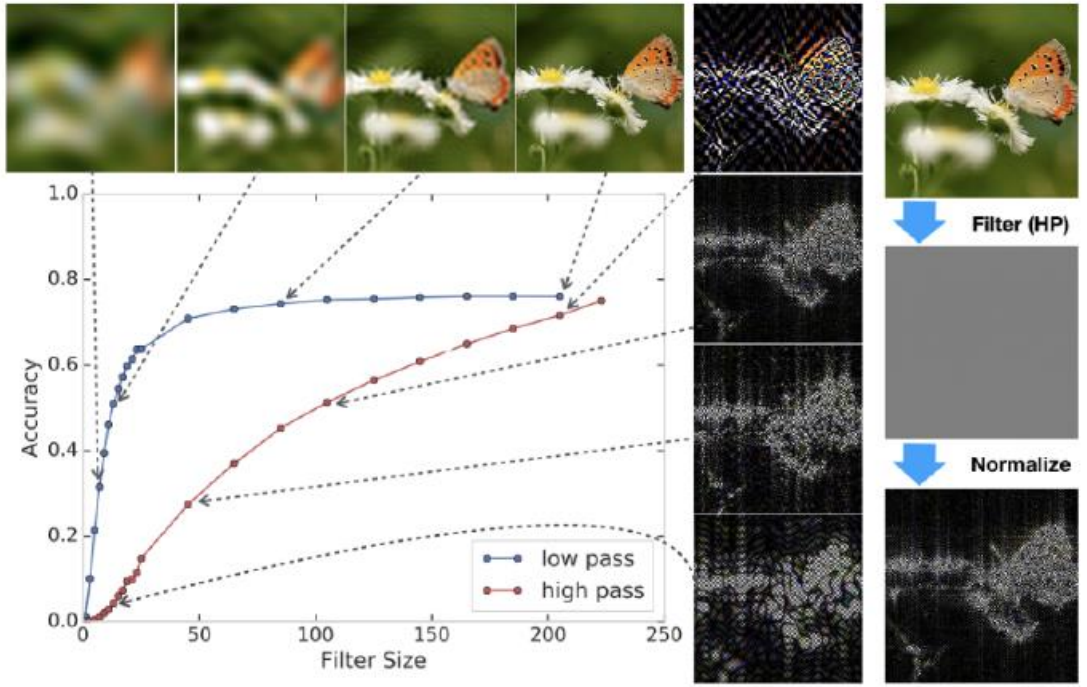


Figure 4.9: Accuracy of a model trained on ImageNet on low-pass and high-pass filtered images, with different filtering intensities, from [90]. The model shows good accuracy even on images that are unrecognizable for humans.

Why do adversarial examples exist ?

Fourier analysis of robustness

Experiments show that models trained with Adversarial Training and gaussian data augmentation become less sensitive to high-frequency perturbations but more sensitive to low-frequency ones.

=> these two techniques make the models rely more on low-frequency information

> *E.g. a model trained with gaussian data augmentation will be more robust to HF perturbations (snow, random noise, etc.) but less robust to LF perturbations such as fog and contrast. Similarly, an adversarially trained model has poorer performances when considering the fog perturbation (LF) than a naturally trained model.*

However, adding LF information during training does not improve robustness to LF perturbations. The authors hypothesize that as natural images are more concentrated in LF, it is harder for the model to become unaware of LF information (thus LF perturbations).

Why do adversarial examples exist ?

Intepretability of saliency maps and robustness to adversarial examples

Etmann et al. [117] investigate the relation between the robustness to adversarial examples and the interpretability of the saliency maps.

Why do adversarial examples exist ?

QuantifiCare

Intepretability of saliency maps and robustness to adversarial examples

Reminder:

Considering an input space of dimension $d = n^2$ (for input images of size $n \times n$ for example), the saliency map is the matrix

$$S(x) = [\nabla_{x_i} F_m(x)]_{i \in [\![i,d]\!]}$$

where $F_m(x)$ is the maximum value of the confidence score vector $F(x)$. The interpretability of the saliency map is represented by the alignment of the saliency map with the input as:

$$\alpha(x) = \frac{|x \cdot S(x)|}{\|S(x)\|_2}$$

The robustness of a model is defined as $\rho(x) = \inf_\epsilon \{\|\epsilon\|_2 \mid M(x+\epsilon) \neq M(x)\}$.

Why do adversarial examples exist ?

QuantifiCare

Intepretability of saliency maps and robustness to adversarial examples

The authors show that for a linear classifier the robustness to adversarial examples and the alignment of the saliency map with the input coincide.

Kaur et al. [118] also notice experimentally that robust models tend to have gradients that are aligned with the human perception

Experiments:
An adversarial example x' from a clean example x targeted towards label y_t is crafted with a large perturbation budget on a standard model, and a model trained with Adversarial Training.
For the robust model, the adversarial example looks like a clean example classified in class y_t, whereas for the standard model, it simply looks like the example x with noisy patterns.

Why do adversarial examples exist ?

Intepretability of saliency maps and robustness to adversarial examples

Kaur et al. [118] also notice experimentally that robust models tend to have gradients that are aligned with the human perception

Why do adversarial examples exist ?

QuantifiCare

Robust models exploit high-level interpretable features

Engstrom et al. [119] show that robust models (here models trained to be robust against adversarial examples with Adversarial Training) learn representations which are more interpretable by humans.

=> (Robust models are shown to extract high-level features which are interpretable by humans.)

A representation that a model has learned for an input x is denoted $g(x)$ and corresponds to the outputs of the penultimate layer.

Experiments:

1) standard models: possible to find inputs which are highly visually dissimilar but have close representations
2) robust models: two images having close representations will be visually similar

Why do adversarial examples exist ?

QuantifiCare

Robust models exploit high-level interpretable features

Objective:

$$x'_1 = x_1 + \text{argmin}_\epsilon \|g(x_1 + \epsilon) - g(x_2)\|_2$$

(adding to x_1 the ϵ that makes $g(x_1 + \epsilon)$ and $g(x_2)$ similar)

Why do adversarial examples exist ?

QuantifiCare

Robust models exploit high-level interpretable features

Standard models:

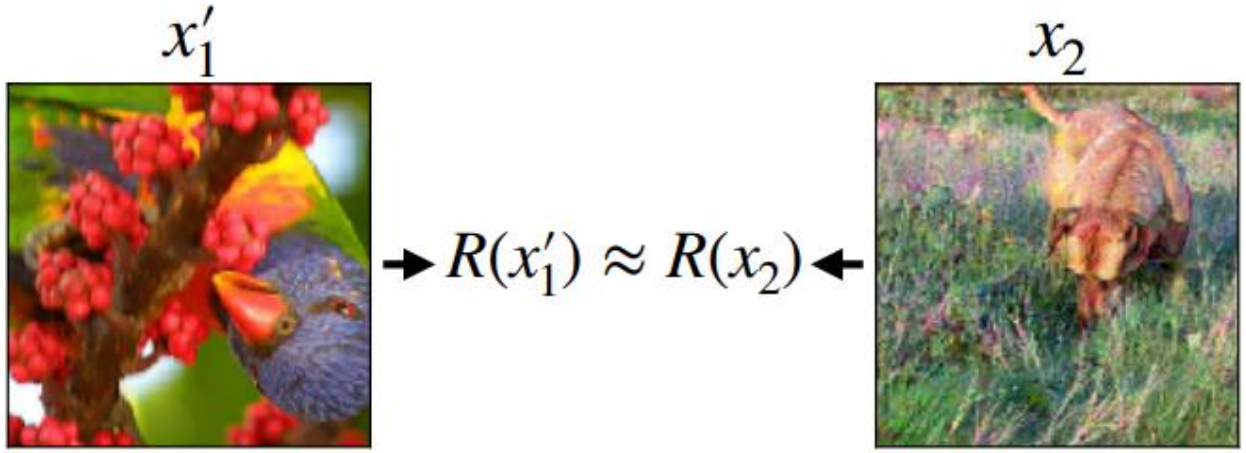


Figure 2: A limitation of standard neural network representations: it is straightforward to construct pairs of images (x_1', x_2) that appear completely different yet map to similar representations.

Robust models:

Hard to minimize both $\|g(x_1 + \epsilon) - g(x_2)\|_2$ and ϵ

Why do adversarial examples exist ?

QuantifiCare

Robust models exploit high-level interpretable features

Figure 3: Visualization of inputs that are mapped to similar representations by models trained on the Restricted ImageNet dataset. *Target* (x_2) & *Source* (x_1): random examples image from the test set; *Robust* and *Standard* (x_1'): result of minimizing the objective (4) to match (in ℓ_2-distance) the representation of the target image starting from the corresponding source image for (*top*): a robust (adversarially trained) and (*bottom*): a standard model respectively. For the robust model, we observe that the resulting images are perceptually similar to the target image in terms of high-level features (even though they do not match it exactly), while for the standard model they often look more similar to the source image which is the seed for the optimization process. Additional results in Appendix B.1, and similar results for ImageNet are in Appendix B.1.4.

Why do adversarial examples exist ?

Robust and non-robust features

Ilyas et al. [57] show that adversarial examples are the consequence of features derived from patterns with a high predictive power, yet these patterns are meaningless to humans and they can be adversarially modified to fool the target classifier.

When training a classifier in order to minimize the empirical risk, no distinction is made between robust and non-robust features. At inference time an adversary modifying non-robust features can then fool the target classifier.

A feature is a function from the input space $\mathcal{X}$ to $\mathbb{R}$ such that the output value of the target classifier is a linear combination of the feature values.
For a neural network, the features values are thus the outputs at the penultimate layer.

Why do adversarial examples exist ?

QuantifiCare

Robust and non-robust features

More formally, considering a binary classification task, f is said to be *ρ-useful* ($\rho > 0$) if it satisfies the following equation:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[y \cdot f(x)\right] > \rho \tag{1}$$

and is *γ-useful robust* if under the worst perturbation δ chosen in a predefined set of allowed perturbations Δ, f stays γ-useful under this perturbation:

$$\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\inf_{\delta\in\Delta} y \cdot f(x+\delta)\right] > \gamma \tag{2}$$

A ρ-useful feature f is said to be a *non-robust feature* if f is not robust for any $\gamma \geq 0$.

Why do adversarial examples exist ?

QuantifiCare

Robust and non-robust features

Experiments:

1) A robust data set mainly composed of robust features.

   starting with an adversarially trained model M_{rob}, for each training set point (x, y), x_r is found by solving $\text{argmin}_z \|g(z) - g(x)\|_2$ (where g is the output function at the penultimate layer of M_{rob}). The starting point of the optimization problem is chosen independently from the true label y in order not to introduce features which are not used by M_{rob}.

2) A data set with mainly non-robust features.

   starting from a clean example x of ground-truth label y, an adversarial example x' targeted towards label y_t is crafted. This new data set is incorrectly labeled according to humans as x' is close to x, but here the adversarial label y_t is considered as the true label. Thus, when y_t is randomly chosen for each example, the robust features of the initial model are now uncorrelated with the label y_t.

Why do adversarial examples exist ?

Robust and non-robust features

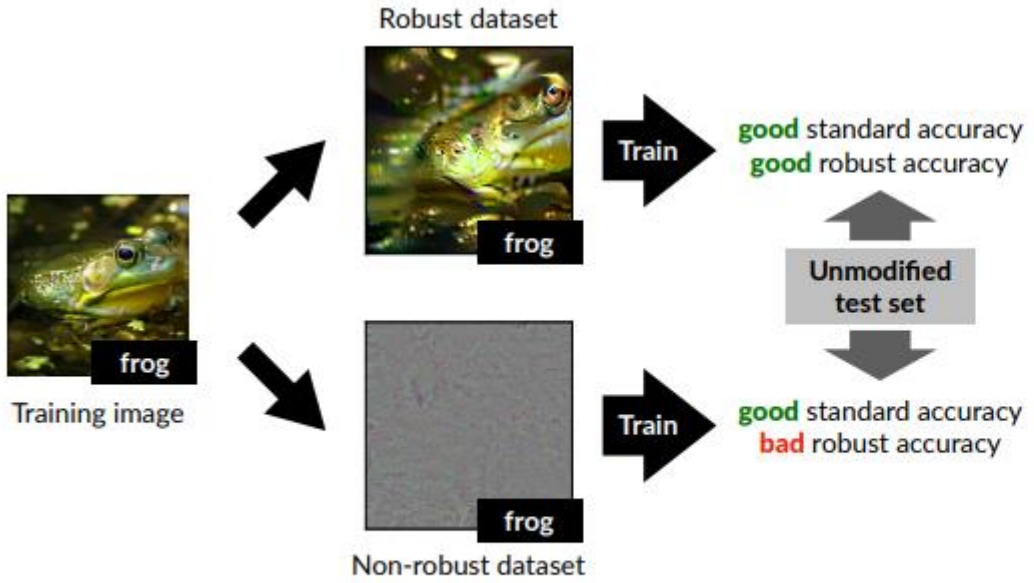
Experiments:
1) Robust data set
=> robustness and accuracy can be reached by discarding non-robust features, i.e. that some non-robust features are responsible for the vulnerability of classifiers to adversarial examples.

2) Non-robust data set
=> models rely on non-robust features at training time.

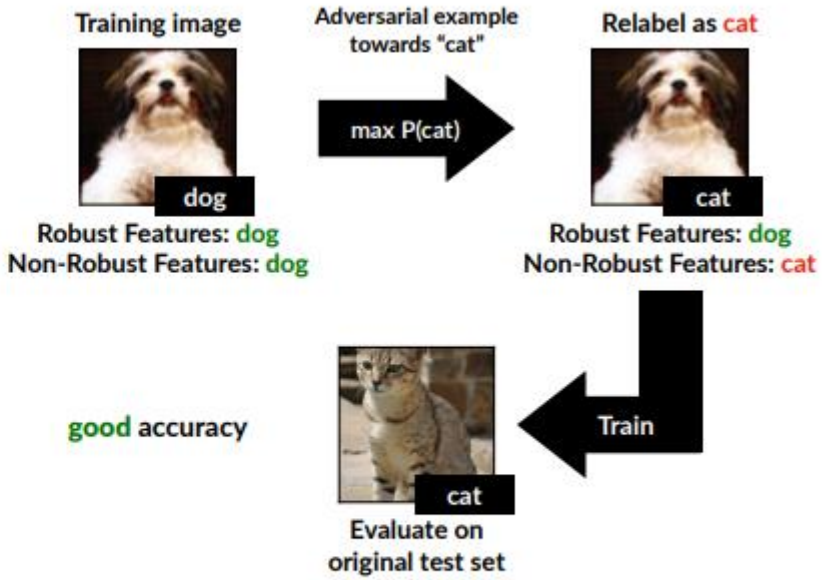
Figure 4.10: Accuracy of models on the CIFAR10 data set, on standard examples (blue) and adversarial examples (red). $\hat{\mathcal{D}}_R$ denotes the robust data set (i.e. with almost only robust features), and $\hat{\mathcal{D}}_{NR}$ denotes the non-robust data set (i.e. with almost only non-robust features), from [57].

Why do adversarial examples exist ?

Robust and non-robust features



(a)



(b)

Attack methods
(White-box setting)

The adversary can compute gradients

Attack methods

Preliminaries

For all the attacks presented here, we denote by $\mathcal{Q}$ the domain in which inputs are considered. As we study image classification tasks, $\mathcal{Q}$ is typically a range in which values of pixels must fit in.

For an image x, we denote by $\text{Clip}(x, \mathcal{Q})$ the clipping operation of every pixel on this image to the domain $\mathcal{Q}$.

Unless specified otherwise, the adversary aims at crafting an adversarial example x' considering a clean input-label pair (x, y).

The attack schemes are only presented in their untargeted version, unless the attack is specifically designed to craft targeted adversarial examples.

Attack methods

Preliminaries

The l_p norm of a vector $u \in \mathbb{R}^d$ for $0 < p < \infty$, denoted by $\|u\|_p$, corresponds to:

$$\|u\|_p = \left(\sum_{i=1}^{d} |u_i|^p \right)^{\frac{1}{p}}$$

The l_∞ norm of a vector $u \in \mathbb{R}^d$, denoted by $\|u\|_\infty$, corresponds to:

$$\|u\|_\infty = \max_{i=1}^{d} |u_i|$$

The l_0 norm of a vector $u \in \mathbb{R}^d$, denoted by $\|u\|_0$, corresponds to:

$$\|u\|_0 = \sum_{i=1}^{d} \mathbb{1}_{\{u_i \neq 0\}}$$

Attack methods

Preliminaries

The l_p ball of center c and radius r, corresponding to $\left\{u \mid \|u - c\|_p \leq r\right\}$, is denoted by $B_p(c, r)$. $B_p(r)$ simply denotes the l_p ball of radius r centered at origin.

We denote by $\text{proj}_{B_p(c,r)}(u)$ the projection of a vector u onto the l_p ball of center c and radius r.

Attack methods

QuantifiCare

L-BFGS algorithm

Szegedy et al. [11] are the first to propose a crafting method in a white-box setting. They consider the objective of finding the smallest l_2 adversarial distortion such that the adversarial example is misclassified.

$$\underset{\epsilon}{\operatorname{argmin}} \ \|\epsilon\|_2$$

$$\text{s.t.} \begin{cases} M_\theta(x+\epsilon) \neq y \\ x+\epsilon \in \mathcal{Q} \end{cases}$$

To solve this problem, the authors search for the minimum c (with a box-constrained L-BFGS algorithm) for which the solution ϵ to the problem satisfies $M_\theta(x+\epsilon) \neq y$.

$$\underset{\epsilon}{\operatorname{argmin}} \ c\,\|\epsilon\|_2 - \mathcal{L}(x+\epsilon, y, M_\theta)$$

$$\text{s.t. } x+\epsilon \in \mathcal{Q}$$

Attack methods

L-BFGS algorithm

Ostrich

Struthio

Camelus

Attack methods

Fast Gradient Sign Method (FGSM)

Goodfellow et al. [73] present the FGSM. In order to craft adversarial examples, it is searched for the worst-case adversarial perturbation in a l_p ball of radius ϵ relatively to the value of the loss function $\mathcal{L}$

$$\underset{\alpha}{\operatorname{argmax}}\ \mathcal{L}(x+\alpha, y, M_\theta)$$
$$\text{s.t} \quad \|\alpha\|_p \leq \epsilon$$

By performing a first order approximation of $\mathcal{L}$ around x,the problems turns to:

$$\underset{\alpha}{\operatorname{argmax}}\ \alpha \cdot \nabla_x \mathcal{L}(x, y, M_\theta)$$
$$\text{s.t} \quad \|\alpha\|_p \leq \epsilon$$

Attack methods

Fast Gradient Sign Method (FGSM)

By performing a first order approximation of $\mathcal{L}$ around x,the problems turns to:

$$\underset{\alpha}{\text{argmax}}\ \alpha \cdot \nabla_x \mathcal{L}(x, y, M_\theta)$$
$$\text{s.t} \quad \|\alpha\|_p \leq \epsilon$$

In their work, the authors only consider the l_∞ norm, and in that case we have $\alpha = \epsilon\,\text{sign}(\nabla_x \mathcal{L}(x, y, M_\theta))$. Therefore, the adversarial example x' is crafted with:

$$x' = x + \epsilon\,\text{sign}(\nabla_x \mathcal{L}(x, y, M_\theta))$$

Attack methods

Fast Gradient Sign Method (FGSM)

This attack can be extended to other l_p norms with $1 \leq p < \infty$. In this case, we have

$$\alpha = \epsilon \frac{\nabla_x \mathcal{L}(x, y, M_\theta)}{\|\nabla_x \mathcal{L}(x, y, M_\theta)\|_p}$$

and then x' is given by:

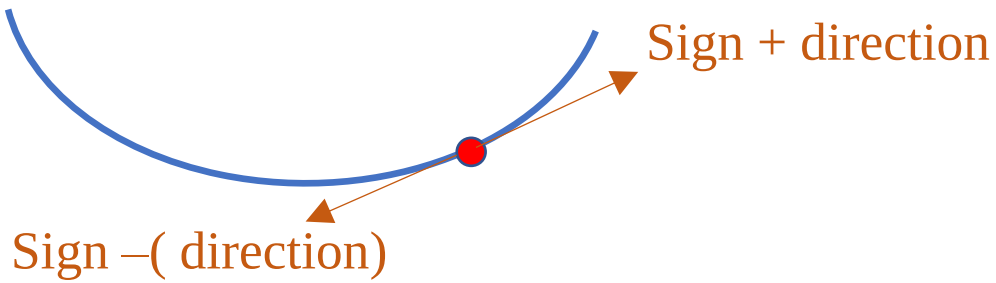
$$x' = x + \epsilon \frac{\nabla_x \mathcal{L}(x, y, M_\theta)}{\|\nabla_x \mathcal{L}(x, y, M_\theta)\|_p}$$

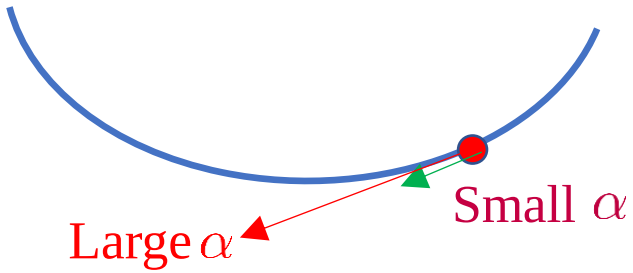
Clipping to ensure that $x' \in \mathcal{Q}$ is eventually performed.

Attack methods

Fast Gradient Sign Method (FGSM)

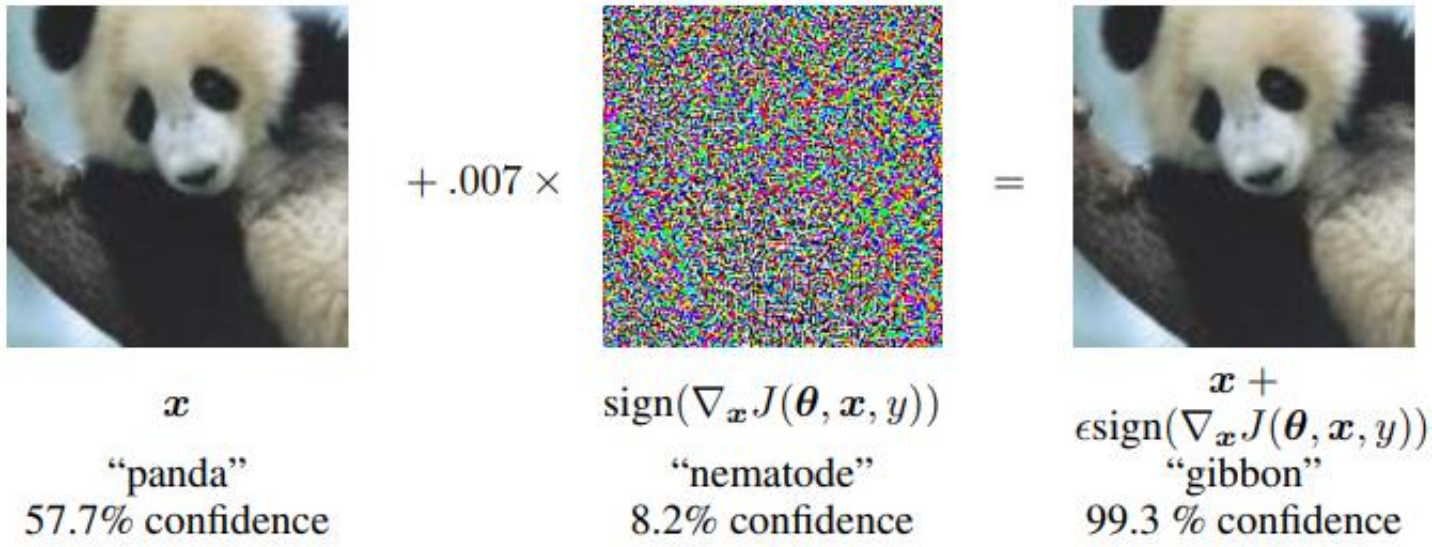
Considering the landscape of the loss function:





Attack methods

Fast Gradient Sign Method (FGSM)

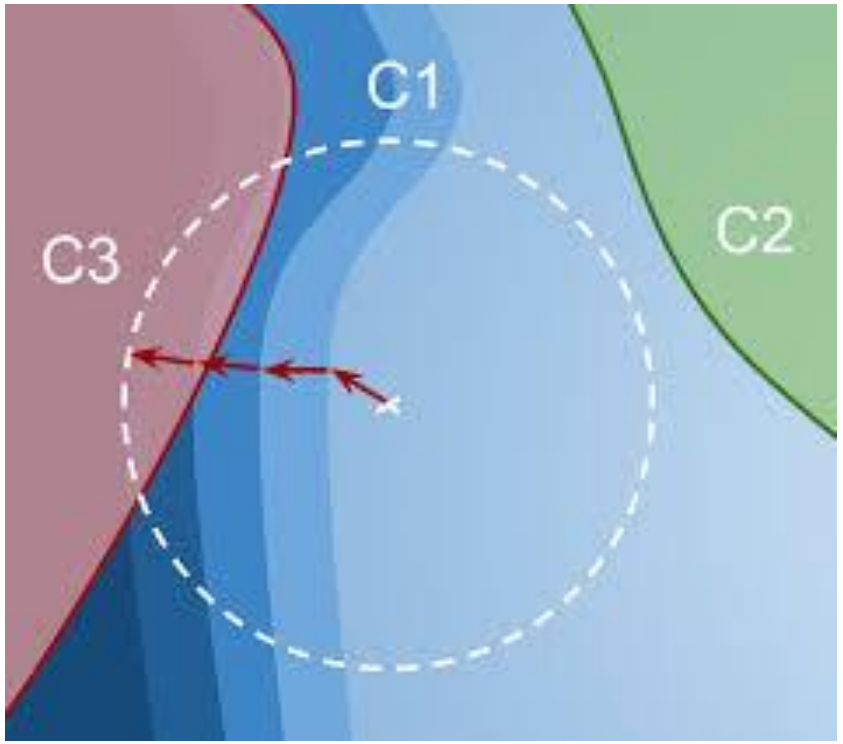


Attack methods

Basic Iterative Method (BIM)

Kurakin et al. [100] extend the FGSM with a multi-step version of it.

This attack scheme consists in iteratively performing the FGSM attack for a given number of iterations, with a step size $\alpha < \epsilon$, projection onto the ball $B_p(x, \epsilon)$ being performed at each step. Clipping to $\mathcal{Q}$ is performed at the end of each step.



Attack methods

Basic Iterative Method (BIM)

The BIM attack can be seen as the projected gradient descent algorithm applied to the optimization problem given in previous slide, apart from the fact that the sign of $\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta)$ is considered instead of only $\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta)$.

Starting point:

$$x^0 = x$$

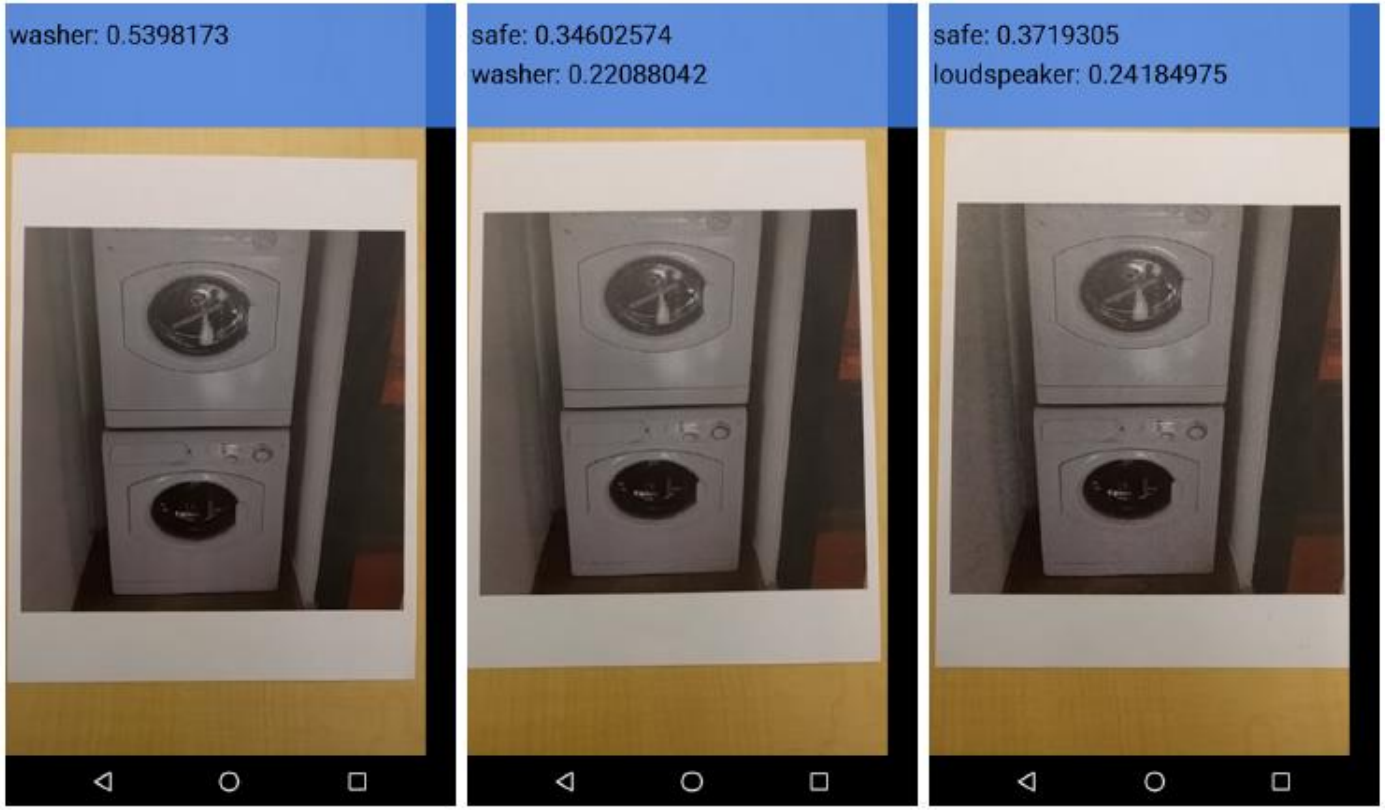
Iterations:

$$\delta^t = \alpha \operatorname{sign}(\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta))$$
$$x^t = \operatorname{proj}_{B_\infty(x,\epsilon)}(x^{t-1} + \delta^t)$$
$$x^t = \operatorname{Clip}(x^t, \mathcal{Q})$$

x^t denotes the example obtained at step t

Attack methods

Basic Iterative Method (BIM)



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

Attack methods

Projected Gradient Descent (PGD)

Madry et al. [75] introduce the PGD, which simply refers to the BIM attack where an initial random r step sampled from a uniform distribution in the range $[-\epsilon, \epsilon]$ is taken.

Experimentally, this step has been shown to allow to escape possible local minima around x.
Moreover, in order to better explore the output space, and find higher values for $\mathcal{L}(x', y, M_\theta)$, the adversary can perform the PGD attack multiple times, with a different starting point each time (thanks to the initial random sampling), and keep only the example inducing the highest $\mathcal{L}(x', y, M_\theta)$ value.

Attack methods

Projected Gradient Descent (PGD)

Starting point:

$$x^0 = x + \eta \qquad \eta \sim \mathcal{U}(-\epsilon, \epsilon)$$

$\mathcal{U}$ denotes the uniform distribution

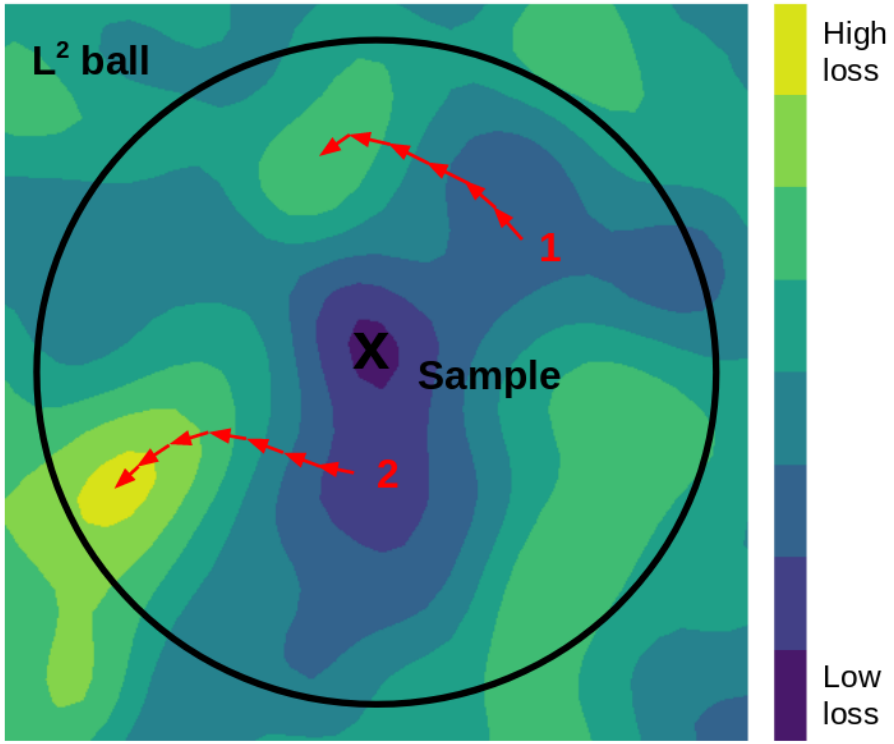
Iterations:

$$\delta^t = \alpha \operatorname{sign}(\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta))$$
$$x^t = \operatorname{proj}_{B_p(x,\epsilon)}(x^{t-1} + \delta^t)$$
$$x^t = \operatorname{Clip}(x^t, \mathcal{Q})$$

x^t denotes the example obtained at step t

Attack methods

Projected Gradient Descent (PGD)



Attack methods

Projected Gradient Descent (PGD)

Sparse l_1 PGD. Tramer et al. [120] propose an enhanced version of the l_1 version of the PGD attack. For the initial PGD attack in its l_1 version, the perturbation δ^t computed at each step t results in a vector with *only one* non-null component being $\operatorname{sign} g_s^t$ with g_s^t denoting the component of $g^t = \nabla_x \mathcal{L}(x^{t-1}, y, M)$ for indice $s = \operatorname{argmax}_i |g_i^t|$.

Therefore, the authors propose a modified version of it where δ^t is set to $\operatorname{sign} g^t$ if $|g^t| > P_q(|g^t|)$, and to 0 otherwise, where $P_q(|g^t|)$ is the q^{th} percentile of $|g^t|$. The smaller the parameter q, the more δ components are likely to be modified. The perturbation δ is then normalized to the l_1 unit-ball (i.e divided by $\|\delta^t\|_1$).

The final perturbation is thus $\alpha \cdot \frac{\delta^t}{\|\delta^t\|_1}$

Attack methods

Projected Gradient Descent (PGD)

Wasserstein PGD. Wong et al. [83] propose a modification of the PGD attack to consider the Wasserstein distance instead of the distance derived from a l_p norm. A threat model based on the Wasserstein distance encompasses modifications such as translation, rotation and scaling better than a threat model based on the l_p norm. At each step of the PGD attack, the projection is performed on a Wasserstein ball and not on a l_p ball.

Attack methods

Projected Gradient Descent (PGD)

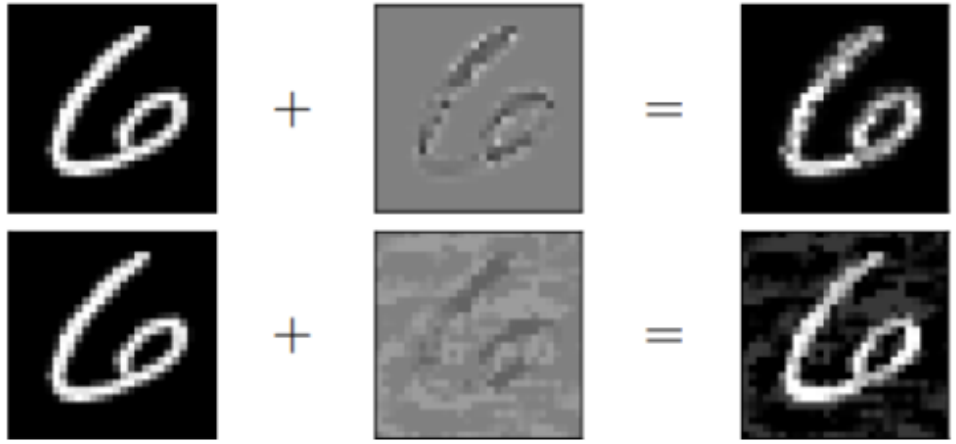


Figure 4.11: Adversarial examples, crafted with Wasserstein PGD (top), and l_∞-PGD (bottom), from [83].

Attack methods

Projected Gradient Descent (PGD)

Auto-PGD. Croce et al. [87] propose Auto-PGD (APGD), an enhanced version of the PGD attack, which only requires to specify the number of iterations and the l_∞ bound for the adversarial perturbation. Specifically, it considers an adaptive step size which takes into account the number of iterations the adversary wants to perform.

Idea:
Allow the step size to decrease when no improvement in the optimization objective occurs, and when the step size decreases, to restart at the best solution found so far.

Attack methods

Projected Gradient Descent (PGD)

This attack has allowed to break numerous defenses !

#	paper	model	source	venue	clean	AA	reported	reduct.
	(Zhang & Wang, 2019)	WideResNet-28-10	available	NeurIPS 19	89.98	36.64	60.6	-23.96
	(Zhang & Xu, 2020)	WideResNet-28-10	available	ICLR 20 R	90.25	36.45	68.7	-32.25
	(Jang et al., 2019)	ResNet20	available	ICCV 19	78.91	34.95	37.40	-2.45
	(Kim & Wang, 2020)	WideResNet-34-10	available	ICLR 20 R	91.51	34.22	57.23	-23.01
	(Moosavi-Dezfooli et al., 2019)	ResNet18	authors	CVPR 19	80.41	33.70	36.3	-2.60
	(Wang & Zhang, 2019)	WideResNet-28-10	available	ICCV 19	92.80	29.35	58.6	-29.25
	(Wang & Zhang, 2019)	WideResNet-28-10	available	ICCV 19	92.82	26.93	66.9	-39.97
	(Mustafa et al., 2019)	ResNet110	available	ICCV 19	89.16	0.28	32.32	-32.04
	(Chan et al., 2020)	WideResNet-34-10	retrained	ICLR 20	93.79	0.26	15.5	-15.24
	(Pang et al., 2020)	ResNet110	available	ICLR 20	93.52	0.00	31.4	-31.40

Attack methods

Carlini-Wagner (CW)

Carlini et al. [105] consider the joint objective of finding an adversarial example and minimizing the l_2 distortion related to this adversarial example

$$\underset{\epsilon}{\text{argmin}} \ \|\epsilon\|_2 + c\,G(x+\epsilon, y)$$

$$\text{s.t.} \quad x + \epsilon \in \mathcal{Q}$$

with:

$$G(x+\epsilon, y) = \max(h_y(x+\epsilon) - \max_{j \neq y} h_j(x+\epsilon), -\kappa)$$

$h_y(x)$ denotes the logit value for the ground-truth indice and κ is a parameter for the desired gap between the logit value for y and the second biggest logit value

Attack methods

Carlini-Wagner (CW)

To solve this optimization problem, the change of variable $x+\epsilon = \frac{1}{2}(\tanh(w)+1)$ is performed to get rid of the box constraint, and the resulting optimization problem is then solved with respect to w with the Adam optimizer.

Moreover, this problem is solved for many values of the constant $c \in \mathbb{R}^+$ found with binary search to find the solution with the lowest l_2 distortion.

Attack methods

Jacobian-based Saliency Map Method (JSMA)

Papernot et al. [121] propose the JSMA method, which is based on the idea of identifying the most sensitive pixels in terms of misclassification, and then modify them.

This attack works in an iterative fashion. As long as the maximum number of iterations is not reached, or that the adversarial distortion, in terms of number of modified pixels does not reach a threshold, two steps are performed at each iteration:

1) the adversary begins by measuring the variation of $F_y(x)$ with respect to a variation of each pixel i. To do so is computed the vector of partial derivatives of the confidence score value for the ground-truth label y, for every pixel i, i.e. compute $\nabla_{x_i} F_y(x)\ \forall i \in [\![1, d]\!]$.
2) the *saliency map* $S(x, y)$ is computed, which allows to target the pixels which will have the more influence for misclassification.

Attack methods

QuantifiCare

Jacobian-based Saliency Map Method (JSMA)

2) the *saliency map* $S(x, y)$ is computed, which allows to target the pixels which will have the more influence for misclassification.

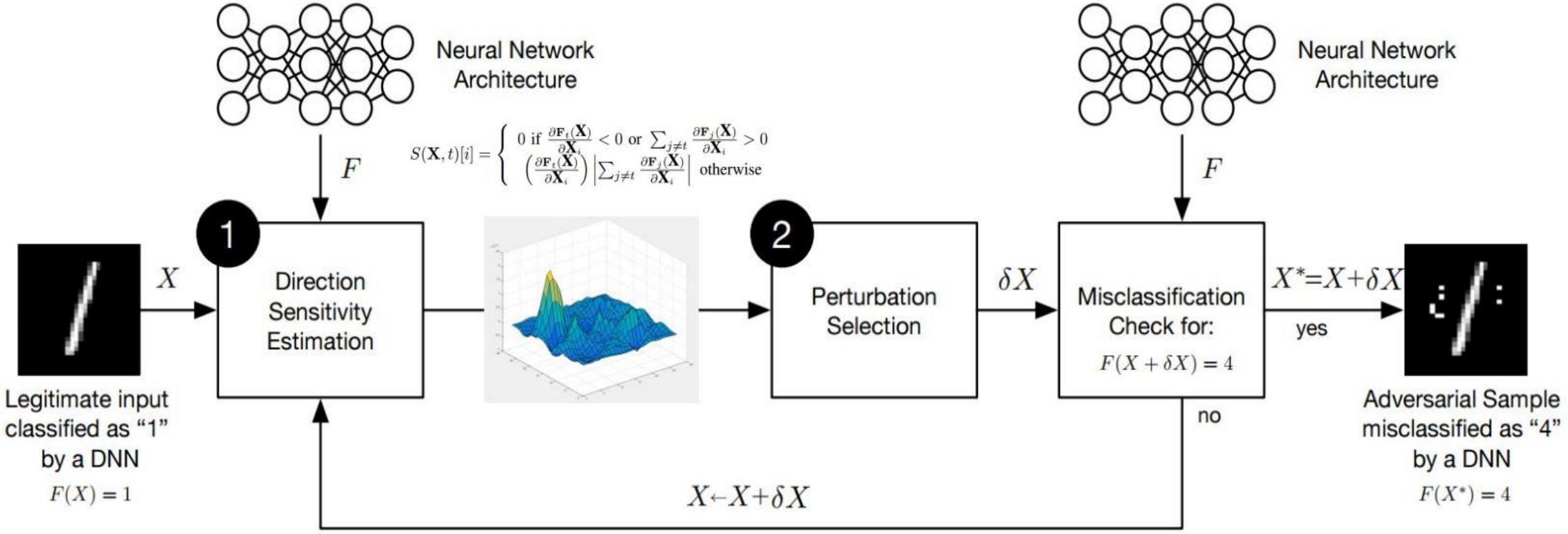
Each component of $S(x, y)$, corresponding to a pixel i, is given by:

$$S_i(x, y) = \begin{cases} 0 \text{ if } \nabla_{x_i} F_y(x) > 0 \text{ or } \sum_{j \neq y} \nabla_{x_j} F_j(x) < 0 \\ |\nabla_{x_i} F_y(x)| \left(\sum_{j \neq y} \nabla_{x_j} F_j(x) \right) \text{ otherwise} \end{cases}$$

Then, the adversary adds a pre-defined perturbation δ to the pixel $i^* = \text{argmax}_i \, S_i(x, y)$. Clipping to the set $\mathcal{Q}$ is eventually performed.

Attack methods

Jacobian-based Saliency Map Method (JSMA)



Attack methods

Decoupling Direction and Norm (DDN)

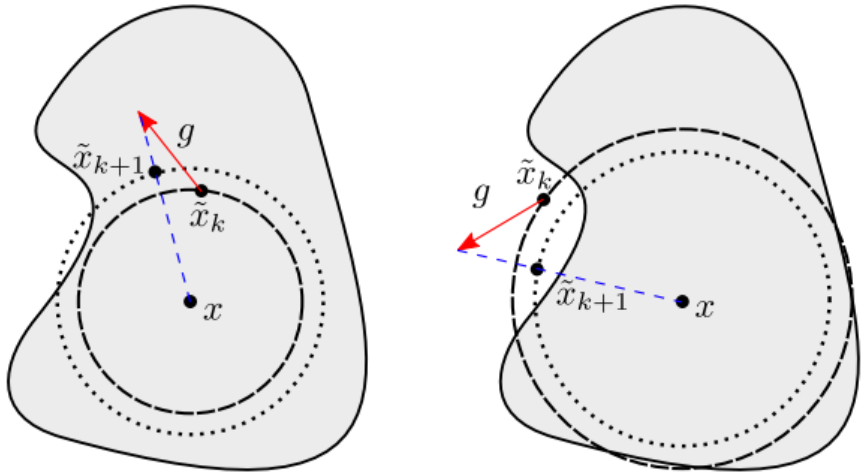
Rony et al. [122] propose the iterative DDN attack.

Contrary to previously presented attacks, the goal is not to search for the minimal adversarial distortion as with the CW attack in its l_2 version, neither to fix a maximum allowed adversarial distortion, as with the FGSM or PGD attack.

Rather, the idea is to project at each step t the resulting example x^t onto a l_2 ball of some radius ϵ^t, and increase or decrease the radius ϵ^t at next step whether the current x^t was found adversarial or not (i.e. is misclassified or not).

Attack methods

Decoupling Direction and Norm (DDN)



(a) $\tilde{x}_k$ not adversarial (b) $\tilde{x}_k$ adversarial

Figure 3: Illustration of an untargeted attack. The shaded area denotes the region of the input space classified as y_{true}. In (a), $\tilde{x}_k$ is still not adversarial, and we increase the norm ϵ_{k+1} for the next iteration, otherwise it is reduced in (b). In both cases, we take a step g starting from the current point $\tilde{x}$, and project back to an ϵ_{k+1}-sphere centered at x.

The perturbation g is proportional to $\nabla_x J(\theta, x_k, y)$ with the true label y

Attack methods

Sparse adversarial examples

Croce et al. [107] propose a method to craft sparse adversarial examples.

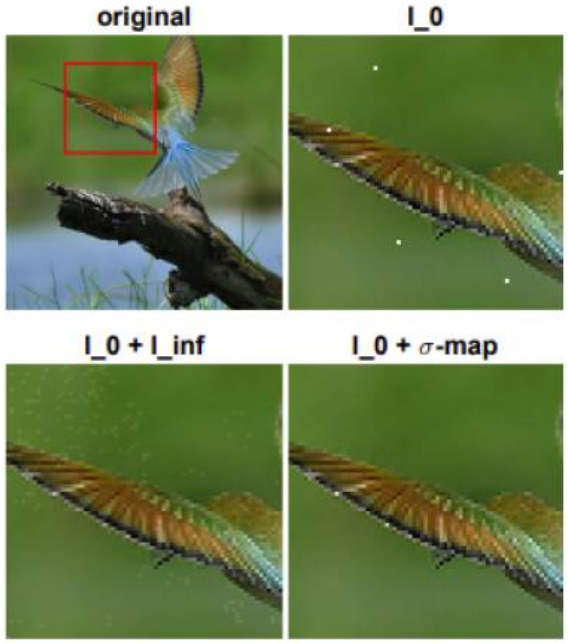


Figure 4.12: Illustration of adversarial examples crafted with the different version of the attack proposed in [107].

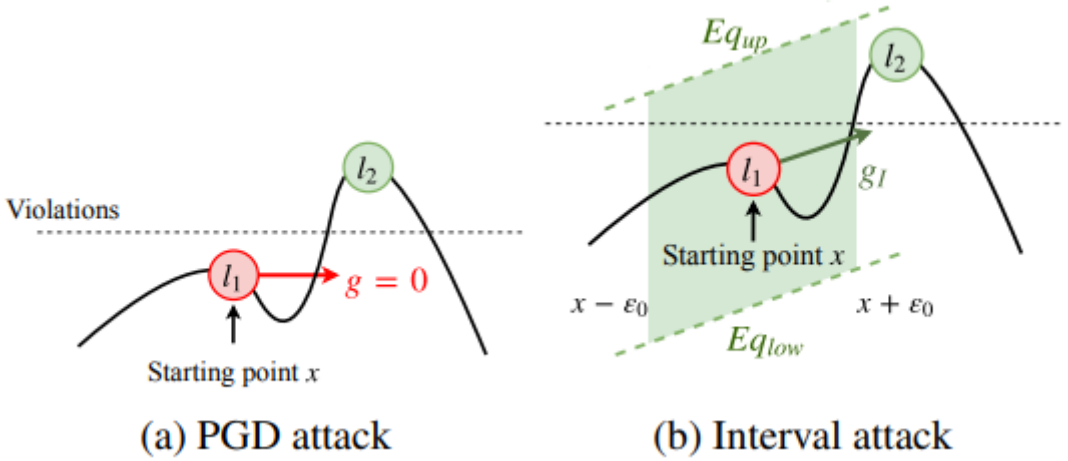
$l_0 + \sigma$-map : a certain number of pixels are modified and the bound for the modification of each pixel depends on image specific characteristics around the pixel, in order to make the perturbed pixel as imperceivable as possible

e.g. if the standard deviation along one axis around the pixel equals 0, the pixel can not be modified as it would be easily spotted

Attack methods

Enhancing gradient-based attacks

Wang et al. [125] propose a way to enhance the choice of the initial point for gradient-based attacks. Starting from a point x_0, the idea is to identify a zone around this point and use sound bound propagation methods to over approximate the output variations in this zone. Then, based on this approximation, the best point in terms of desired output variations is found. This point is then used as the initial point for the gradient-based attack.

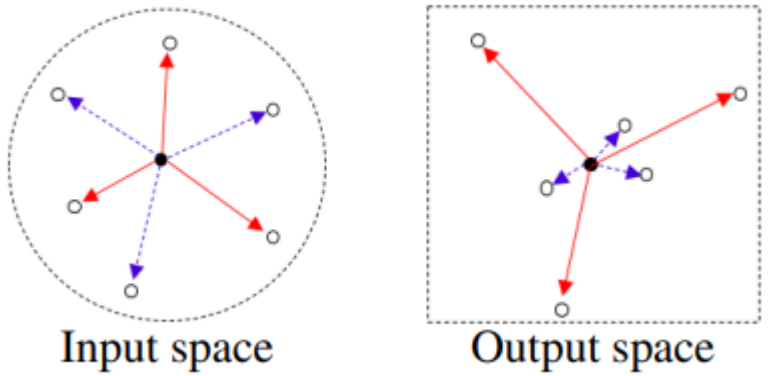


(a) PGD attack (b) Interval attack

Attack methods

Enhancing gradient-based attacks

Tashiro et al. [126] propose Output Diversified Initialization (ODI) to improve the initialization step for attacks which use a certain number of restarts, with a different initialization at each restart. These restarts are performed in order to increase diversity among beginning points of the attack, with the aim of better exploring the adversarial subspace of the output space, and then select the best adversarial example. The idea of the ODI scheme is to leverage diversity in the output space rather than in the input space. The motivation comes from the fact that as models tend to be very non-linear, considering diversity in the input space does not necessarily induce diversity in the output space.



Attack methods (Black-box setting: score-based and decision-based attacks)

The adversary can not compute gradients

Attack methods

Zeroth-order optimization

Chen et al. [127] propose a score-based attack (an adversary supposed only to know the confidence score vector $F(x)$), which uses a gradient estimation method.

The loss to be optimized to craft adversarial examples is the one used for the CW attack where the logits function h is replaced by the score function F:

$$\underset{\epsilon}{\operatorname{argmin}} \ \|\epsilon\|_2 + c\, G_F(x+\epsilon, y)$$

$$\text{s.t.} \quad x + \epsilon \in \mathcal{Q}$$

with:

$$G_F(x+\epsilon, y) = \max(F_y(x+\epsilon) - \max_{j \neq y} F_j(x+\epsilon), -\kappa)$$

Attack methods

Zeroth-order optimization

The optimization procedure is done with the Adam optimizer where the following approximation is used for the derivative of F with respect to x_i (the i[th] element of x):

$$\nabla_{x_i} F(x) \approx \frac{F(x + he_i) - F(x - he_i)}{2h}$$

e_i is a vector with the i[th] component set to 1, and all other components valuing 0.

The computation of the derivative of F with respect to x_i has to be done d times for each iteration of the optimization algorithm, and requires two evaluations of the score function F at each time, it can quickly become very expensive in terms of computation costs to reach convergence.

Solutions:
1) solve the optimization problem in a stochastic way, only updating one pixel at each minimization step. To
2) focus on pixels near the main objects on the pictures, or use dimension reduction

Attack methods

Simultaneous Perturbation Stochastic Approximation (SPSA)

Uesato et al. [79] also propose a score-based attack scheme exploiting a gradient estimation method.

The adversary is supposed to have access to the logits $h(x)$, and the adversarial example is searched by solving the following optimization problem:

$$\underset{x'}{\operatorname{argmin}}\ h_y(x') - \max_{j \neq y} h_j(x')$$
$$\text{s.t. } \|x' - x\|_\infty \leq \epsilon$$

Attack methods

QuantifiCare

Simultaneous Perturbation Stochastic Approximation (SPSA)

The optimization problem is solved with SPSA:

At each step t with current example x^t, n vectors $v_1, \ldots, v_n$ of dimension d are considered. For each vector v_j ($j \in [\![1, n]\!]$), its i^{th} component $v_{j,i}$ is drawn from a Rademacher distribution (a discrete probability distribution: $v_{i,j}$ has 50% chance of being $+1$, and 50% chance of being -1).

Then is computed $g_j = \frac{(h(x^t + \delta v_j) - h(x^t - \delta v_j))}{2\delta}$, where δ is a perturbation size.

The example is then updated as $x^{t+1} = \text{proj}_{B_\infty(x,\epsilon)}(x^t - \alpha \sum_{j=1}^{n} g_j)$, where α is a step size.

Attack methods

Simple Black-box Attack (SimBA)

Guo et al. [129] propose a gradient estimation free score-based attack to craft adversarial examples, where the adversary only has access to the confidence score vector.

Idea:
Pick orthogonal directions in a set without replacement, updating the adversarial perturbation by one of these directions if it lowers the confidence score for the predicted label.

The orthogonality of potential directions allows not to search in directions cancelling each other or making the adversarial perturbation grow too much.

Attack methods

Boundary Attack

Brendel et al. [130] propose a decision-based attack method (the adversary is only allowed to retrieve the output label of the target model M).

Idea:
Consider at start an example x^0 classified in a different class than x (i.e. $M(x^0) \neq M(x)$), and to make steps towards x while staying adversarial, adapting the step procedure at each iteration.

Iterative procedure, at each iteration (if the example stays adversarial):
- Orthogonal step: draw $\eta \sim \mathcal{N}(0, 1)$, scale it, perturb x^t with it and project on a sphere with some given radius s.t. $d(x^0, x^{t-1}) = d(x^0, x^{t-1} + \eta_{scaled})$
- Take a small step towards x^0 to get x^t

Attack methods

Boundary Attack

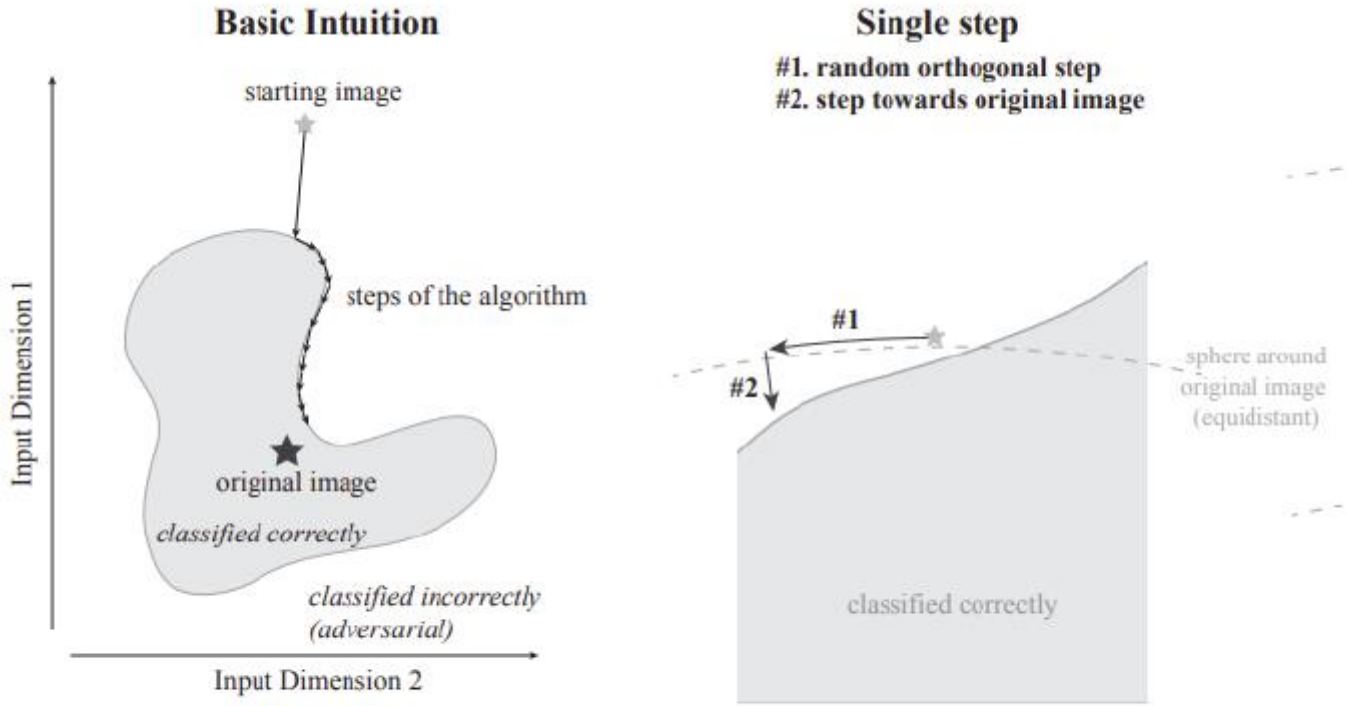


Figure 4.13: Illustration of the intuition of the Boundary Attack, from [130].

Attack methods

Hop Skip Jump Attack (HSJA)

Chen et al. [108] propose a decision-based attack method, which relies on a gradient estimation which is only valid on the decision boundary.

Idea:
Considering a clean example x^0, classified in a different class than x, this attack scheme works in an iterative manner to get closer to x while staying adversarial.

Each step involves three main components:

1) Firstly, binary-search is performed to reach the decision boundary
2) Gradient estimation at the decision boundary is performed.
3) It is searched for a step size, so that taking a step in the estimated gradient direction keeps the current example adversarial.

Attack methods

Hop Skip Jump Attack (HSJA)

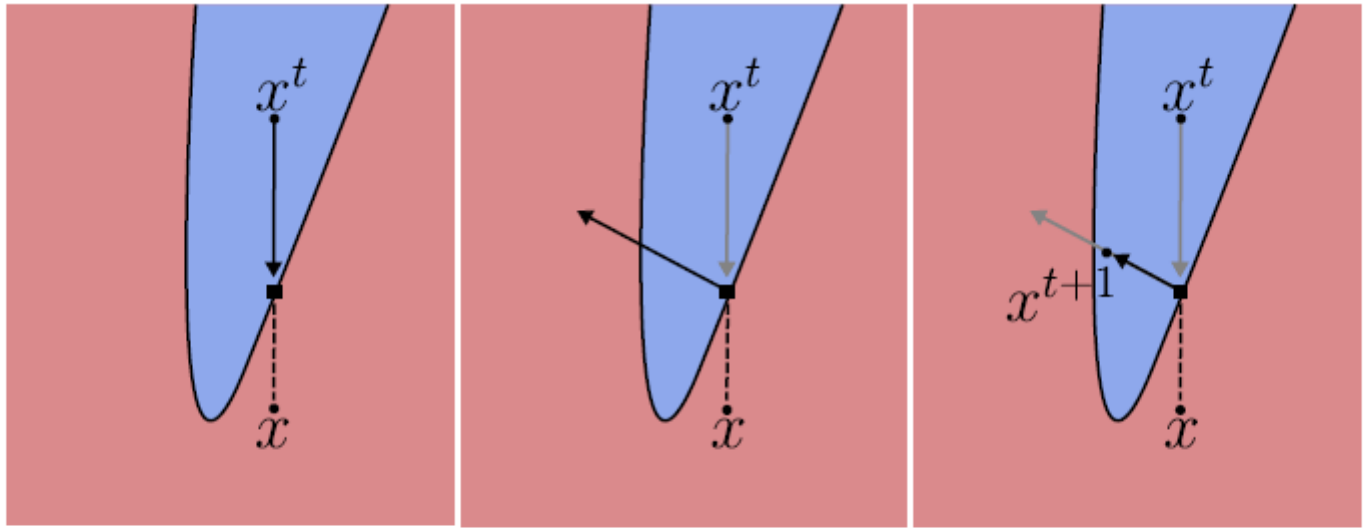


Figure 4.14: Illustration of the intuition of the Hop Skip Jump Attack, from [108]. (Left) Line Search to find the boundary. (Middle) Gradient estimation. (Right) Step size search.

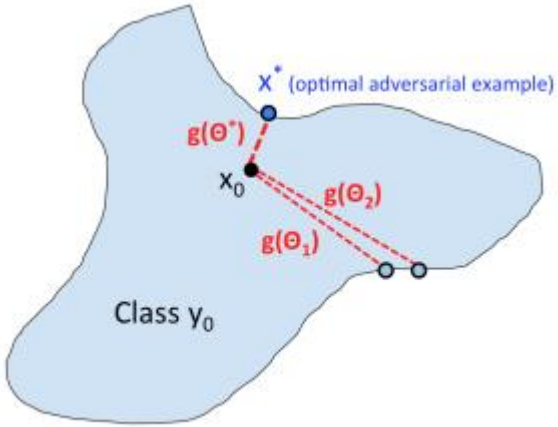
Attack methods

Optimization-based Approach

Cheng et al. [132] propose a decision-based (only the output label) attack that formulates the adversarial example search with an optimization problem that can be solved with a gradient-free optimization method.

$$\underset{\theta}{\text{argmin}} \; g(\theta)$$

$$\text{s.t. } g(\theta) = \underset{\lambda}{\text{argmin}} \left(M(x + \lambda \frac{\theta}{\|\theta\|_2}) \neq y \right)$$

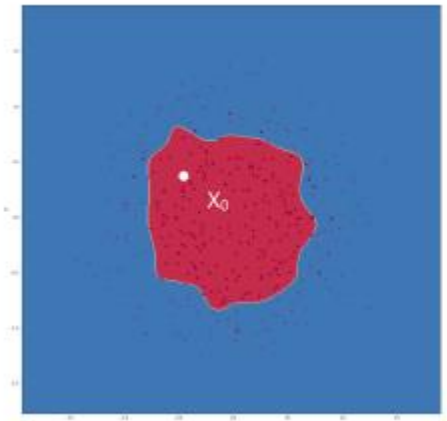


Denoting θ^* the solution to this problem, θ^* is the direction towards the closest adversarial example, and $g(\theta^*)$ is thus the shortest distance towards an adversarial example.

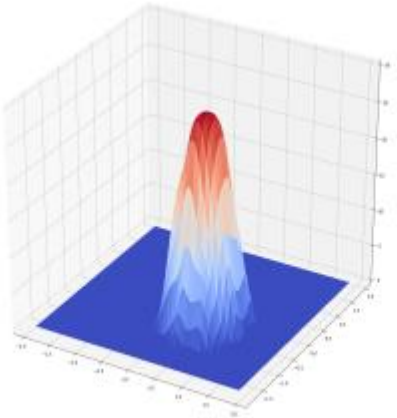
Attack methods

Optimization-based Approach

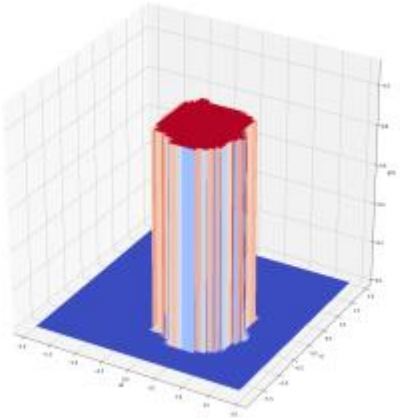
The authors show experimentally that $g(\theta)$ is continuous even if the decision function f is not continuous.



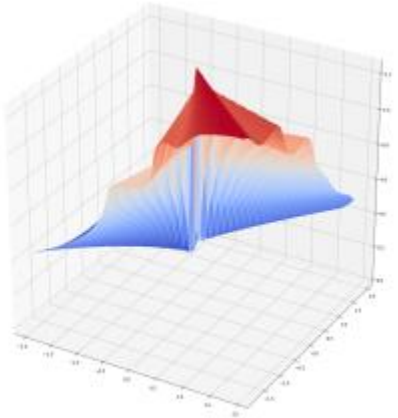
1.



2.



3.



4.

1. Decision boundary

2. Loss with logits is continuous with respect to inputs 3. Loss with labels is not continuous with respect to inputs

4. $g(\theta)$ is continuous

Attack methods

Optimization-based Approach

Objective:

$$\underset{\theta}{\text{argmin}}\ g(\theta)$$

$$\text{s.t. } g(\theta) = \underset{\lambda}{\text{argmin}} \left(M(x + \lambda \frac{\theta}{\|\theta\|_2}) \neq y \right)$$

The derivative of g with respect to θ is approximated by $\hat{g} = \frac{g(\theta+\beta u)-g(\theta)}{\beta} u$ where β is a parameter whose value is fixed to $\beta = 0.005$ and u is a gaussian vector

Update rule:

$$\theta \leftarrow \theta - \eta \hat{g}$$

Attack methods

Optimization-based Approach

For a given θ value, how to find $g(\theta)$?

i.e. how to find the smallest $\lambda > 0$ such that $f(x + \lambda \frac{\theta}{\|\theta\|}) \neq y$?

Firstly, a fine-grained-search is performed: values $f(x+0.01\lambda), f(x+2\times0.01\lambda), ...$ are tested, and we stop at i such as $f(x + (i-1) * 0.01\lambda) = y$ and $f(x + (i) * 0.01\lambda) \neq y$.

Secondly, a binary search is performed between $i-1$ and i to refine the value of $g(\theta)$.

Attack methods

Efficient Combinatorial Optimization (ECO)

Moon et al. [133] also propose a method to craft adversarial examples for an adversary in a black-box setting having access to the values of the loss function. The search for an adversarial example is considered as a discrete optimization problem, which is then solved in a greedy way without any gradient estimation.

Motivation:
Experimental evidence than the adversarial perturbations computed with the l_∞ version of the PGD attack are predominantly found on the vertices of the l_∞ ball.

=> For a l_∞ budget of ϵ, the adversarial perturbations mostly values $-\epsilon$ or ϵ

Attack methods

Efficient Combinatorial Optimization (ECO)

Based on this motivation:

FGSM, BIM, PGD objective:

$$\max_{x'} \mathcal{L}(x, y, M)$$
$$\text{s.t. } \|x' - x\|_\infty \leq \epsilon$$

ECO objective:

$$\max_{x'} \mathcal{L}(x, y, M)$$
$$\text{s.t. } x' - x \in \{-\epsilon, \epsilon\}^d$$

Attack methods

Efficient Combinatorial Optimization (ECO)

Reformulation:

$$\max_{\mathcal{S} \subset \mathcal{V}} F(\mathcal{S})$$

$$\text{s.t. } F(\mathcal{S}) = \mathcal{L}\left(x + \epsilon \sum_{i \in \mathcal{S}} e_i - \epsilon \sum_{i \in \mathcal{V} \setminus \mathcal{S}} e_i, y, M\right)$$

$\mathcal{V}$ denotes the set of all pixel locations
$\mathcal{S}$ denotes the set of pixels modified by $+\epsilon$
$\mathcal{V} \setminus \mathcal{S}$ denotes the set of pixels modified by $-\epsilon$
e_i denotes the i^{th} standard basis vector.

Attack methods

Efficient Combinatorial Optimization (ECO)

Solving:

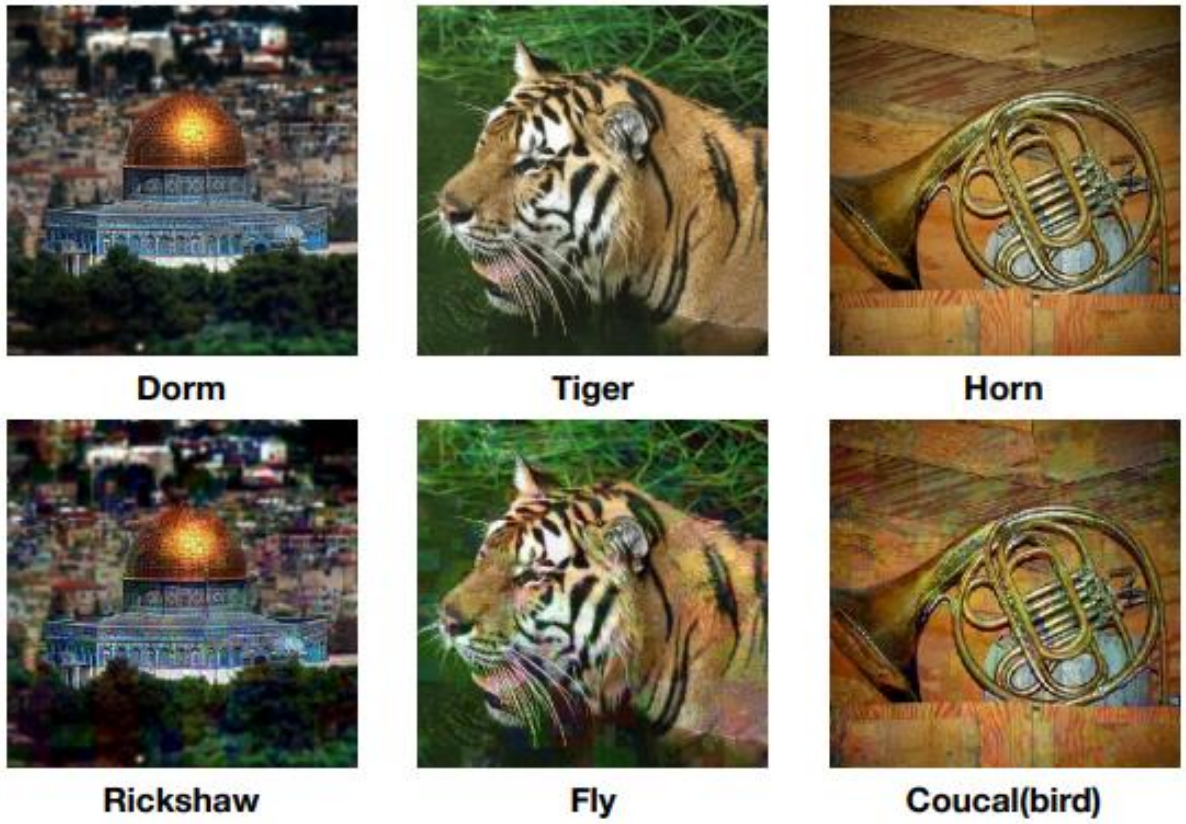
The main challenge in solving this optimization problem is the fact that finding the set $\mathcal{S}$ is NP-hard.
However, the authors show that one can obtain an approximate solution with greedy search algorithms.

Lazy-Greedy algorithm:

Alternatively adding pixels of $\mathcal{S}$ and removing pixels from $\mathcal{S}$ based on a defined rule, until convergence.

Attack methods

Efficient Combinatorial Optimization (ECO)



Attack methods

Rotations and translations

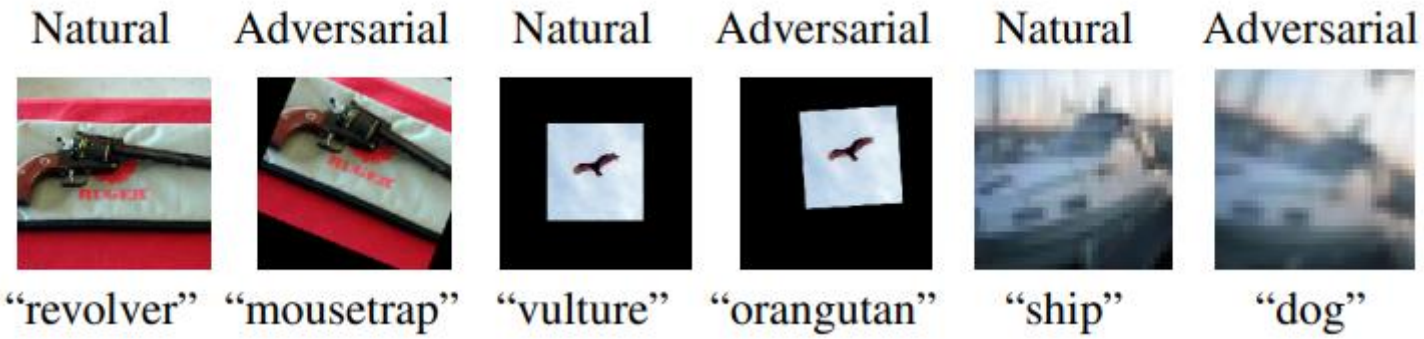
Engstrom et al. [60] propose attack procedures to craft adversarial examples exploiting rotations and translations of clean examples, where the adversary only needs to know the confidence score vector.

Idea:

Considering the value $x_{i,j}$ of a clean example for pixel at position (i, j), the pixel value of the adversarial example at the same position, $x'_{i,j}$ is built by performing a rotation of angle θ and then a translation of δ_u (resp. δ_v) in the direction for the x-axis (resp. y-axis).

Attack methods

Rotations and translations



Attack methods

Colorfool: adversarial colorization

Shamsabadi et al. [61] propose a decision-based attack, where the adversary modifies colors of a clean example to turn it into an adversarial one.

Specificity:
It separates sensitive regions, where large color changes seem unnatural (e.g. face color, sky, etc.), with non-sensitive regions, which can undergo larger color changes (e.g. walls, ground, etc.).

After separating sensitive from non-sensitive regions, the former are converted in the lab color space.
Then, adversarial perturbations in the form of natural color-changes are applied to the non-sensitive regions (multiple trials until an adversarial example is found).

Attack methods

Colorfool: adversarial colorization

Figure 4.16: Illustration of the colorfool attack, with the clean image (left), and its adversarial counterpart (right), from [61]. The background color (non-sensitive region) is largely modified, while the skin color (very sensitive region) is very subtly changed.

Attack methods

Colorfool: adversarial colorization

Su et al. [62] propose a score-based attack method (the adversary only needs to know the softmax output values) to craft adversarial examples while modifying only one pixel.

Idea:
Leverage differential evolution

Starting from a set of candidate solutions.
During each iteration another set of candidate solutions (children) is generated according to the current population (parents). Then the children are compared with their corresponding parents, surviving if they are more fitted (possess higher fitness value) than their parents.
In such a way, only comparing the parent and his child, the goal of keeping diversity and improving fitness values can be simultaneously achieved

Attack methods

Colorfool: adversarial colorization

Idea:
Leverage differential evolution

One candidate solution contains a fixed number of perturbations, with each perturbation corresponding to the perturbation added to one specific pixel. The population is composed of many candidate solutions.

At each iteration, from each perturbation of each candidate solution (the parent) is created another perturbation (the child) with a deterministic formula. The parent or the child is kept for the next iteration depending on which of the two is more fitted (i.e. it reaches better the adversarial goal than another).

The number of iterations is fixed and an early-stopping criterion is fixed in case a perturbation allows the attack to succeed.

Attack methods

Colorfool: adversarial colorization

Attack methods

Colorfool: adversarial colorization

Hosseini et al. [63] propose a decision-based way to craft semantic adversarial examples, i.e unrestricted adversarial examples in terms of l_p norms but semantically similar to clean images they are crafted from.

HUV values (Hue, Saturation and Value) for a color :
Three parameters to describe a color in the way the human vision perceives it.

hue is the color valuing in $[0, 1]$ in a cyclic way $(h(0) = h(1))$
$saturation$ values in $[0, 1]$ with 0 being a gray-scale image and 1 is the most colorful
$value$ is the brightness

x_h, x_s and x_v denote respectively the hue, saturation and value of an image x.

Attack methods

Colorfool: adversarial colorization

Considering a clean image x, craft an adversarial example x' with the only constraint being that $x_v = x'_v$ (same value for the clean and adversarial example, ensuring that objects in the image stay the same).

Minimal saturation shift to avoid gray-scale or too colorful images

=>

$$\min |\delta_s|$$
$$s.t \quad x'_h = (x_h + \delta_h) \, mod \, 1$$
$$x'_s = clip(x_s + \delta_s, 0, 1)$$
$$x'_v = x_v$$
$$M(x') \neq M(x)$$

δ_h, δ_s and δ_v denote respectively the shift in hue, saturation and value.

Attack methods

Colorfool: adversarial colorization

Solving the problem:

Iterative procedure:

Begin with null shift and repeating for $i \in [1, N]$ iterations $\delta_h = \delta_h + \mathcal{U}(0, 1)$ and $\delta_s = \delta_s + \mathcal{U}(\frac{-i}{N}, \frac{i}{N})$, performing proper clipping to respect the constraint and stopping if an adversarial example is found

(smaller updates are performed to δ_s to try to minimize $|\delta_s|$)

Attack methods

Colorfool: adversarial colorization

Attack methods
(Black-box setting: transferability)

Attack methods (transferability)

QuantifiCare

Transferability

Adversary in a black-box setting:
- Score-based attacks
- Decision-based attacks

Limitations:

1) Number of queries can be prohibitive

Pay-per-query system => very costly attack

2) Numerous queries (thousands) can be easily flagged as suspicious

Detection system => account deleted/reported

Attack methods (transferability)

Transferability

In view of these difficulties, the adversary can leverage the transferability of adversarial perturbations

(introduced by Papernot et al. [64])

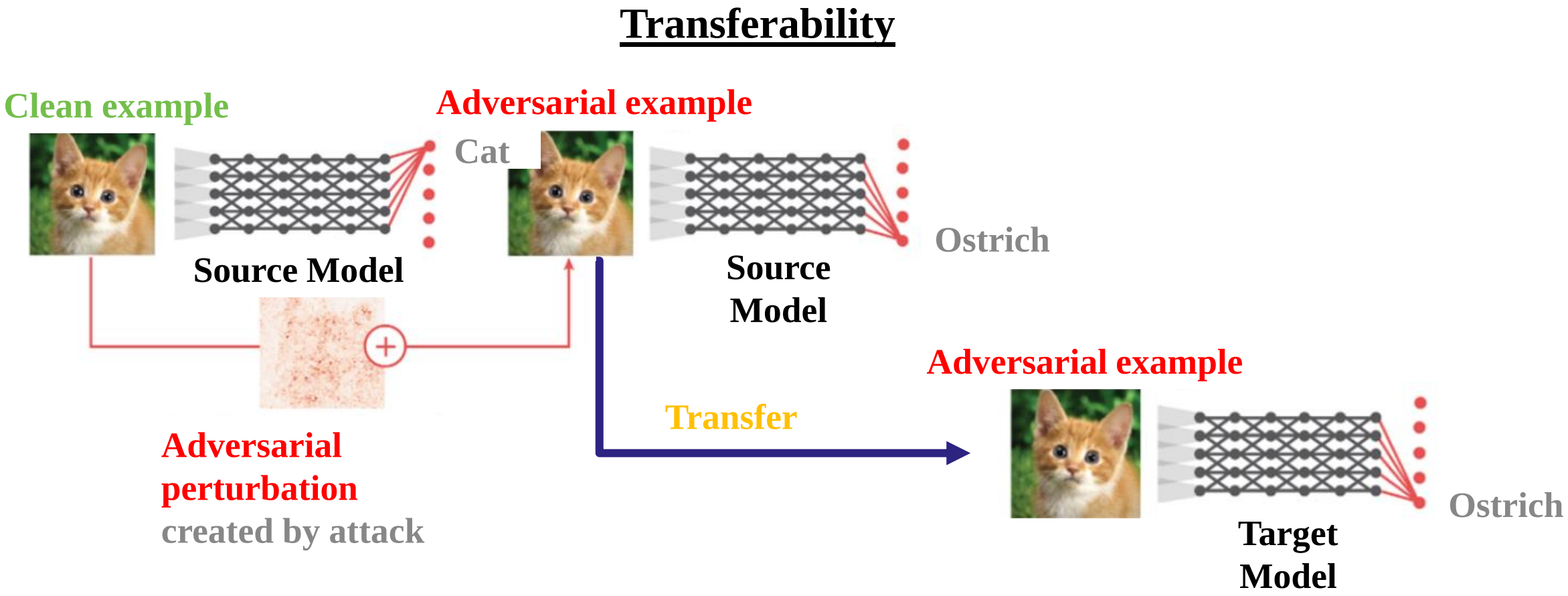
Principle:

Considering the target model M, an adversary can train a substitute (source) model M_s for the same classification task and take advantage of this model to attack the target model M

How ?

An adversarial example x' crafted on M_s ($M_s(x') \neq M_s(x)$) can also fool M ($M(x') \neq M(x)$)

Attack methods (transferability)



Attack methods (transferability)

Momentum Iterative Method (MIM)

Dong et al. [137] propose a method to increase the transferability of adversarial examples crafted with gradient-based attacks.

Idea:
Add a momentum term in the optimization procedure

Indeed, remembering at optimization step the precedent update allows to have more consistent (i.e. in the same direction) updates throughout the attack.

Attack methods (transferability)

Momentum Iterative Method (MIM)

FMGS, BIM, PGD procedure:

$$\underset{x'}{\text{argmax}} \ \mathcal{L}(x', y, M_\theta)$$
$$\text{s.t} \quad \|x' - x\|_p \leq \epsilon$$

At each step:

$$x^t = x^{t-1} + \alpha \ \text{sign}\, g^t \qquad \text{with:} \qquad g^t = \nabla_x \mathcal{L}(x^{t-1}, y, M)$$

MIM:

$$g^t = \mu g^{t-1} + \frac{\nabla_x \mathcal{L}(x^{t-1}, y, M)}{\|\nabla_x \mathcal{L}(x^{t-1}, y, M)\|_1}$$

Attack methods (transferability)

Momentum Iterative Method (MIM)

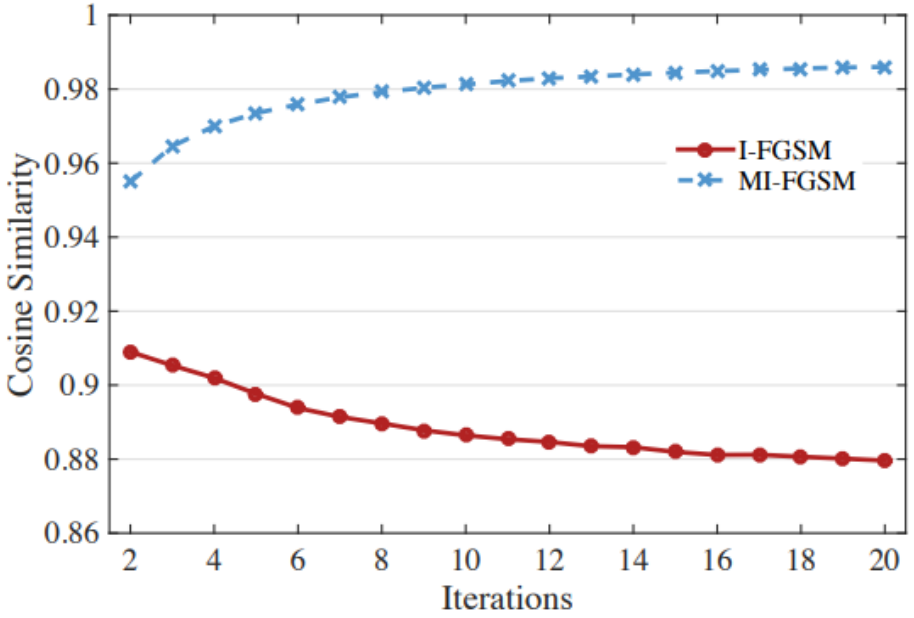
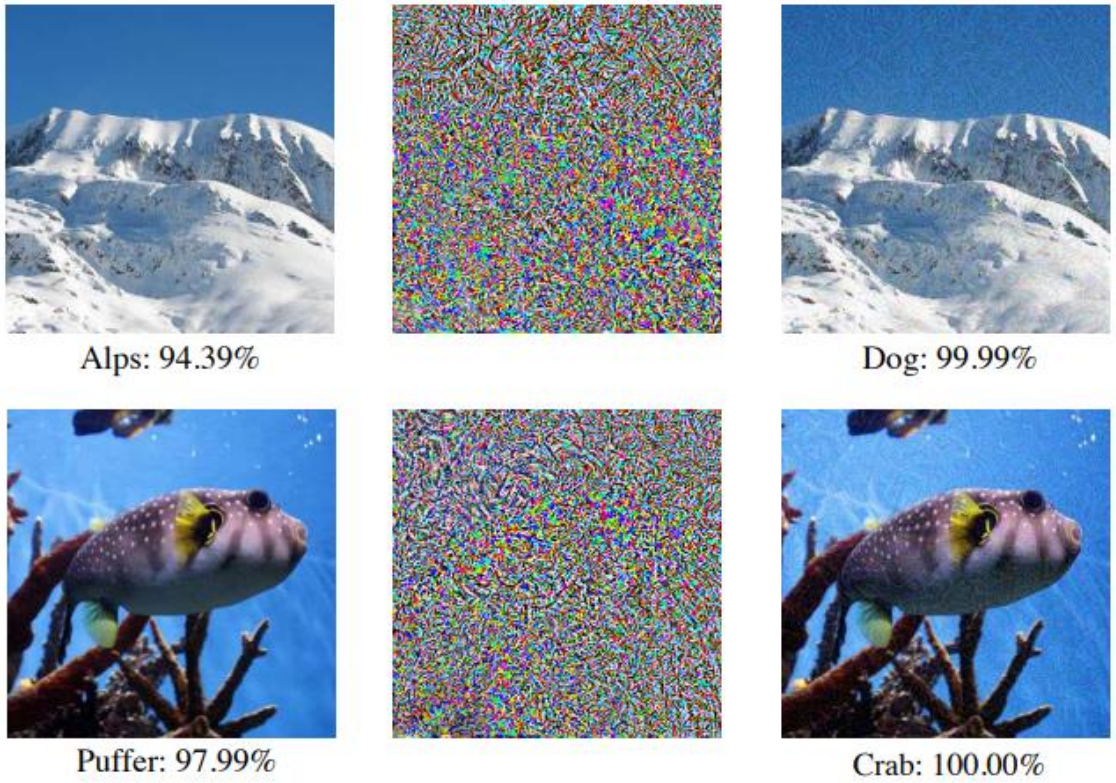


Figure 4. The cosine similarity of two successive perturbations in I-FGSM and MI-FGSM when attacking Inc-v3 model. The results are averaged over 1000 images.

Attack methods (transferability)

Momentum Iterative Method (MIM)



Attack methods (transferability)

Translation Invariant Attack

Dong et al. [138] propose to increase the transferability of adversarial examples crafted with a gradient-based attack by accounting for many translated versions of the input during the optimization procedure.

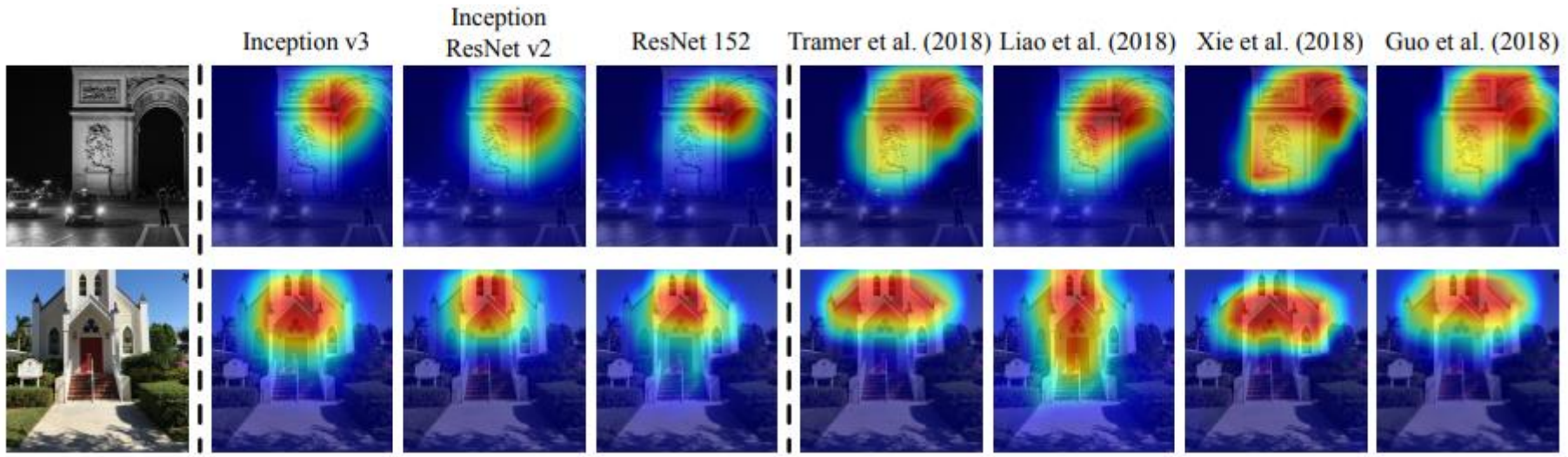
Idea:
Source and target models do not rely on the same region of the image to predict the label, and therefore that crafted adversarial examples tend to overfit to the source model's gradients and discriminative regions.

Considering many translated version of an input for gradient computation could then allow to alleviate this issue.

Attack methods (transferability)

Translation Invariant Attack

Different discriminative regions for different models:



Attack methods (transferability)

Translation Invariant Attack

Same optimization objective as FMGS, BIM, PGD etc.

$$\underset{x'}{\text{argmax}}\ \mathcal{L}(x', y, M_\theta)$$
$$\text{s.t} \quad \|x' - x\|_p \leq \epsilon$$

Extension with shitfted versions of the input:

$$\underset{x'}{\text{argmax}} \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x'), y)$$
$$\text{s.t. } \|x' - x\|_\infty \leq \epsilon$$

$i, j \in [\![-k, k]\!]$ with k a maximum number of pixels to shift
$T_{i,j}(x)$ the shifted version of x by i (resp. j) pixels along the x (resp. y) axis
$w_{i,j}$ coefficients.

Attack methods (transferability)

Translation Invariant Attack

Extension with shitfted versions of the input:

$$\underset{x'}{\operatorname{argmax}} \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x'), y)$$
$$\text{s.t. } \|x' - x\|_\infty \leq \epsilon$$

$i, j \in [\![-k, k]\!]$ with k a maximum number of pixels to shift
$T_{i,j}(x)$ the shifted version of x by i (resp. j) pixels along the x (resp. y) axis
$w_{i,j}$ coefficients.

Solving this optimization problem requires performing $(2k+1)^2$ gradient calculations at each step !

Attack methods (transferability)

Translation Invariant Attack

Solving the optimization problem requires performing $(2k+1)^2$ gradient calculations at each step !

To alleviate this problem, use an approximation:

$$\nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=T_{i,j}(\hat{x})} \simeq \nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=\hat{x}}$$

Translation-invariance property of CNNS
+ images shifted by 10 pixels at most

Attack methods (transferability)

QuantifiCare

Translation Invariant Attack

Solving the optimization problem requires performing $(2k+1)^2$ gradient calculations at each step !

To alleviate this problem:

$$\nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=T_{i,j}(\hat{x})} \simeq \nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=\hat{x}}$$

then:

$$\nabla_x \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x), y)\Big|_{x=\hat{x}} = W \circledast \nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=\hat{x}}$$

$\circledast$: convolution operation

W: kernel matrix of size $(2k+1) \times (2k+1)$

Attack methods (transferability)

Translation Invariant Attack

Finally:

$$\underset{x'}{\operatorname{argmax}} \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x'), y) \quad \text{s.t. } \|x' - x\|_\infty \leq \epsilon$$

$i, j \in [\![-k, k]\!]$ with k a maximum number of pixels to shift
$T_{i,j}(x)$ the shifted version of x by i (resp. j) pixels along the x (resp. y) axis
$w_{i,j}$ coefficients.

with: $$\nabla_x \sum_{i,j} w_{i,j} L(\theta, T_{i,j}(x), y)\Big|_{x=\hat{x}} = W \circledast \nabla_x \mathcal{L}(x, y, M_S)\Big|_{x=\hat{x}}$$

=> One gradient calculation convolved with a kernel matrix (e.g. gaussian kernel)

Attack methods (transferability)

Translation Invariant Attack

Example (PGD with Translation Invariant Attack):

Starting point:

$$x^0 = x + \eta \qquad \eta \sim \mathcal{U}(-\epsilon, \epsilon)$$

$\mathcal{U}$ denotes the uniform distribution

Iterations:

PGD

$$\delta^t = \alpha \operatorname{sign}(\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta))$$
$$x^t = \operatorname{proj}_{B_p(x,\epsilon)}(x^{t-1} + \delta^t)$$
$$x^t = \operatorname{Clip}(x^t, \mathcal{Q})$$

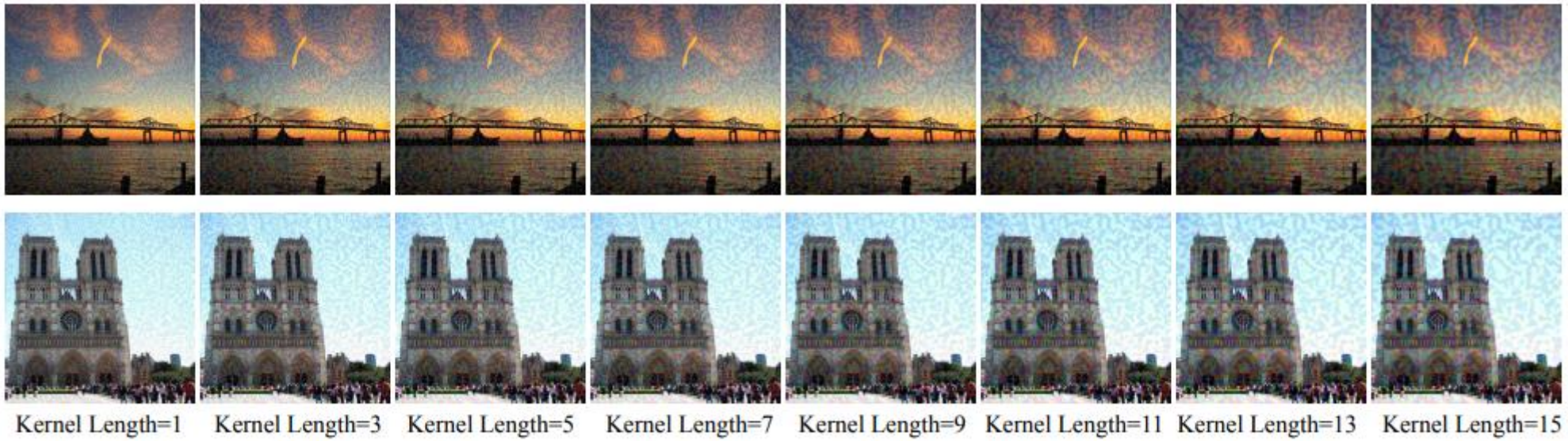
PGD with Translation Invariant Attack

$$\delta^t = \alpha \operatorname{sign}(W \circledast \nabla_x \mathcal{L}(x^{t-1}, y, M_\theta))$$
$$x^t = \operatorname{proj}_{B_p(x,\epsilon)}(x^{t-1} + \delta^t)$$
$$x^t = \operatorname{Clip}(x^t, \mathcal{Q})$$

Attack methods (transferability)

Translation Invariant Attack

Gaussian kernel.



Bigger kernel => smoother perturbations

Attack methods (transferability)

Translation Invariant Attack

Attack methods (transferability)

Improving transferability with input diversity

Xie et al. [139] propose to decrease the overfitting of adversarial examples to the source network they are crafted on.

Idea:
Use data augmentation with spatially transformed inputs during the optimization procedure.

Practically:
At each gradient computation, the input is transformed with probability p with image resizing or random padding.

Attack methods (transferability)

Intermediate Level Attack (ILA)

Huang et al. [140] propose the Intermediate Level Attack (ILA) to increase the transferability of an already crafted adversarial example.

Objective:

Given a hidden layer l of the source model with its output function F_l, and an existing adversarial example x', the attack method aims at maximizing the perturbation at layer l, while keeping the original adversarial direction.

Attack methods (transferability)

Intermediate Level Attack (ILA)

Objective:

Given a hidden layer l of the source model with its output function F_l, and an existing adversarial example x', the attack method aims at maximizing the perturbation at layer l, while keeping the original adversarial direction.

How ?

Maximizing $\Delta(y_l^{''}) = F_l(x'') - F_l(x)$ with respect to x'', while not straying too far from the original adversarial direction given by $\Delta(y_l^{'}) = F_l(x') - F_l(x)$.

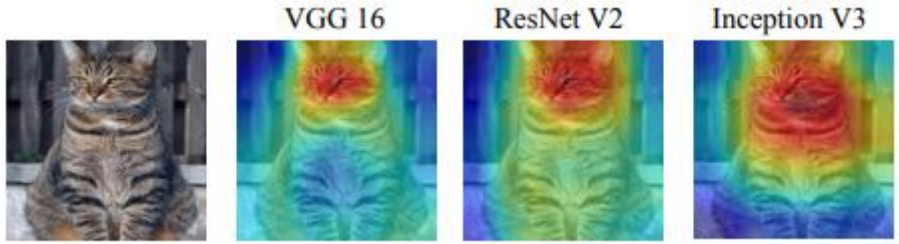
=> Maximize $-\Delta(y_l^{''}) \cdot \Delta(y_l^{'})$ with respect to x''.

Attack methods (transferability)

Intermediate Level Attack (ILA)

Wu et al. [141] propose a method to enhance the transferability of adversarial examples.

Motivation:
It is observed that some features (internal representations in a neural network model) are commonly used by different models with different architectures, and that therefore focusing more on these critical features would lead to a better transferability.



Different models rely on the face region of the cat

Attack methods (transferability)

Intermediate Level Attack (ILA)

Importance of a feature map c of size $m \times n$ at layer k with respect to target class y_t:

$$a_c^k[y_t] = \frac{1}{Z} \sum_m \sum_n \frac{\partial F_{y_t}}{\partial A_k^c[m,n]}(x)$$

$A_k^c[m,n](x)$: value at the feature map c of layer k for input x
Z: normalization constant such that $a_c^k \in [-1,1]$

Attention map for label y_t at laker k :

$$H_k^{y_t}(x) = ReLU(\sum_c a_c^k[t] A_k^c)$$

(Important features for label y_t at laker k)

Attack methods (transferability)

Intermediate Level Attack (ILA)

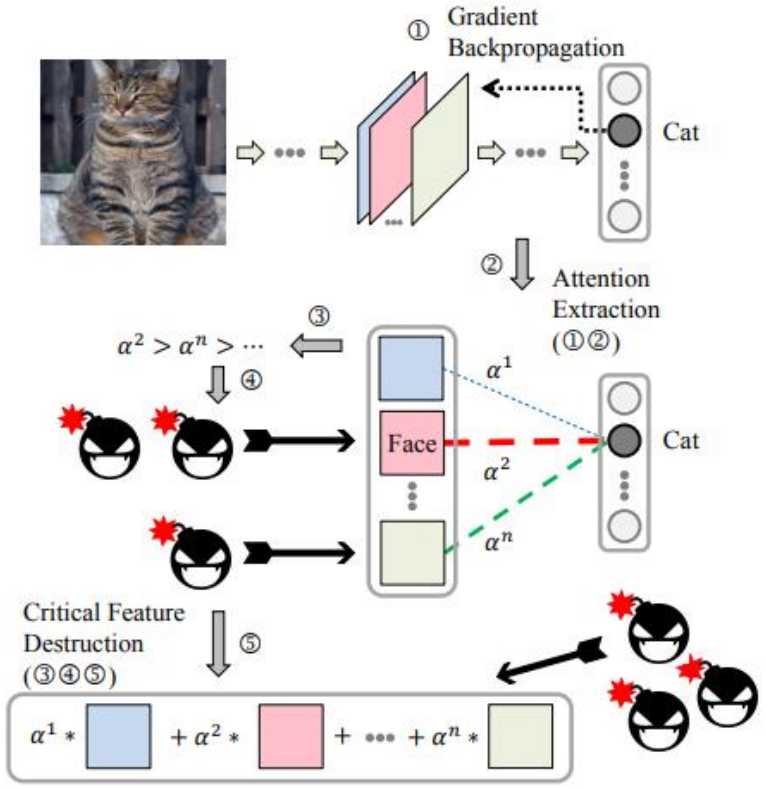
Attack (targeted version):

$$\underset{x'}{\operatorname{argmax}}\ \mathcal{L}(x', y_t, M_S) + \lambda \sum_{k=1}^{K} \left\| H_k^{y_t}(x') - H_k^{y_t}(x) \right\|_2$$
$$s.t.\ \left\| x' - x \right\|_\infty \leq \epsilon$$

Second term: ensure focus on features with a positive influence on the target class

Attack methods (transferability)

Intermediate Level Attack (ILA)



Attack methods (transferability)

Direction Aggregated Attack (DA)

Huang et al. [142] propose a method to enhance the transferability of adversarial examples crafted.

Idea:
Replace each gradient computation (direction) with an aggregation of directions for examples perturbed with gaussian noise.

By doing so, it is hoped:

1) *relieve overfitting to the source network, allowing to find more meaningful directions to fool the target model.*
2) *reduce oscillations between successive gradients directions*

Attack methods (transferability)

Direction Aggregated Attack (DA)

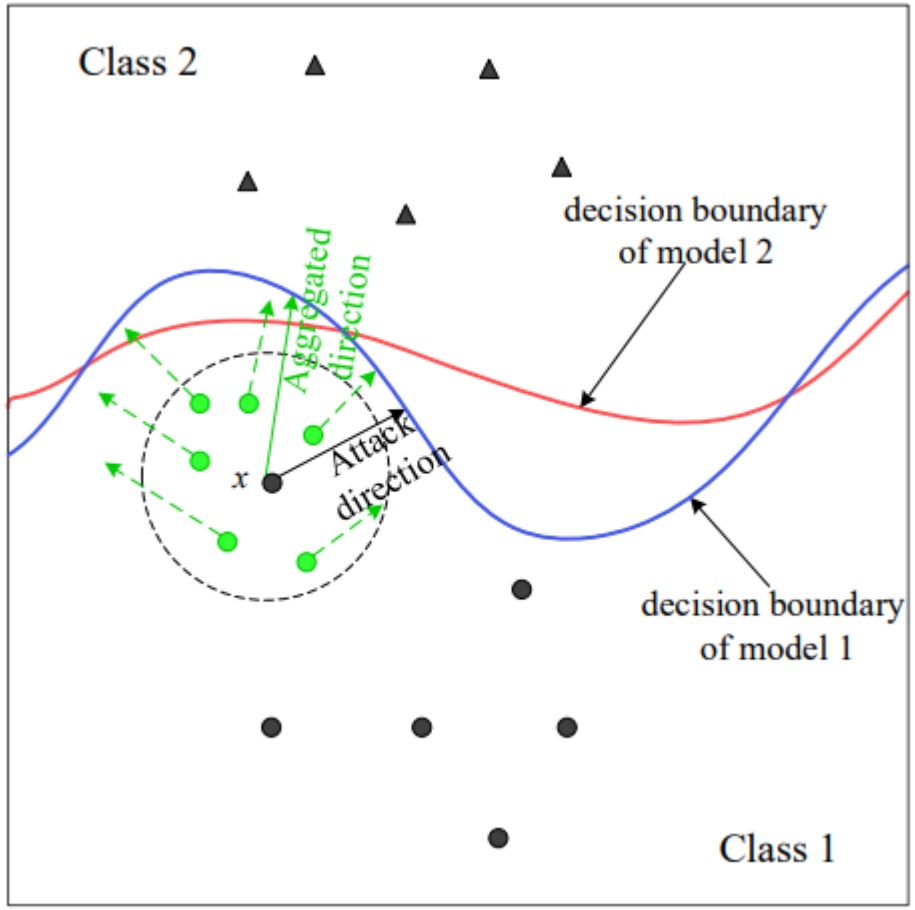
Classical perturbation:

$$\nabla_x \mathcal{L}(x, y, M_S)$$

Replaced by:

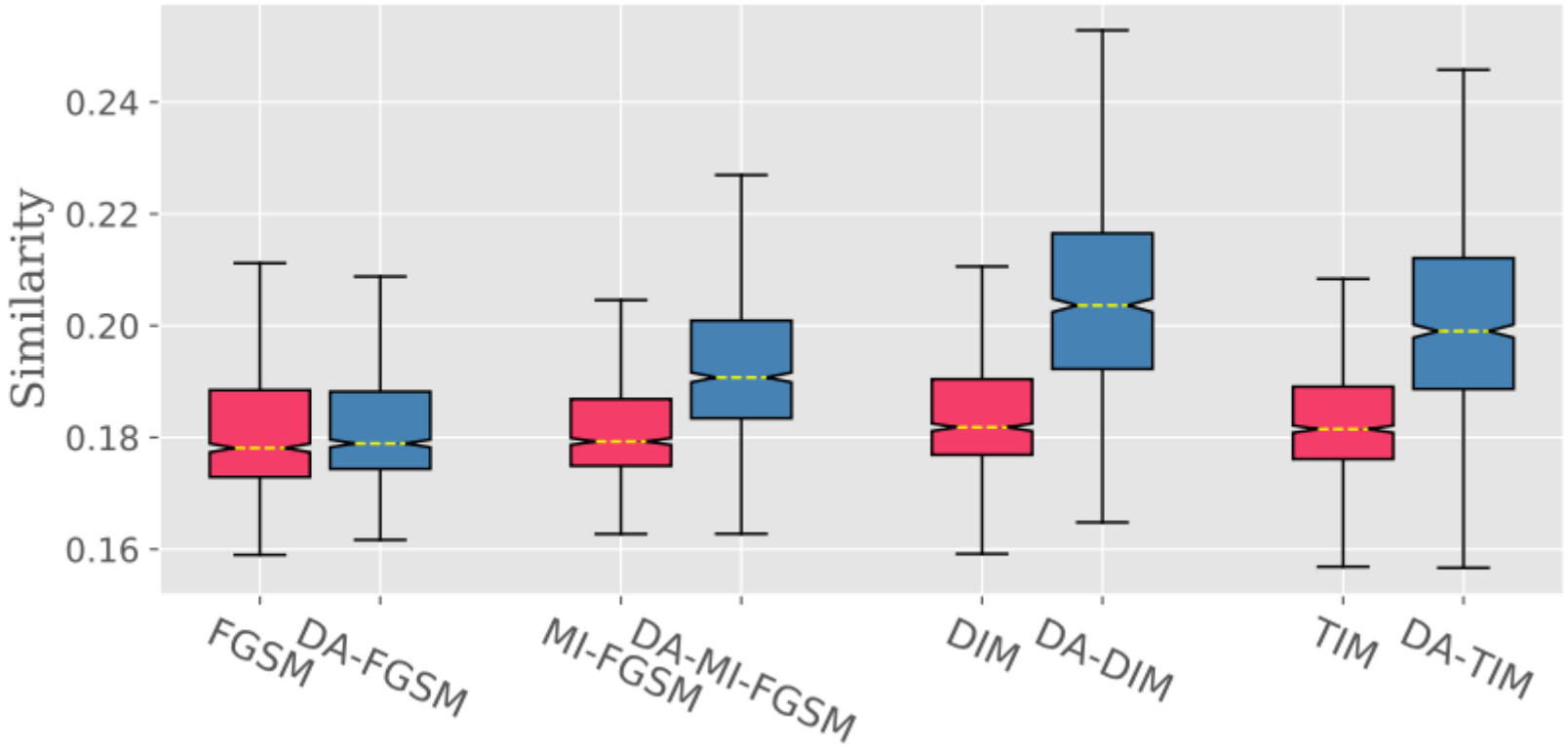
$$\sum_{i=1}^{T} \nabla_x \mathcal{L}(x + \epsilon_i, y, M_S)$$

with: $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$



Attack methods (transferability)

Direction Aggregated Attack (DA)



Attack methods (transferability)

Transferable Adversarial Perturbations (TAP)

Zhou et al. [143] propose a method to craft transferable adversarial examples, leveraging a specific loss function.

Idea:
Use the feature representation in intermediate layers, and the reducing of the variations of the adversarial perturbation.

A loss function with three specific components is used.

$$\mathcal{L}_{TAP} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3$$

Attack methods (transferability)

Transferable Adversarial Perturbations (TAP)

A loss function with three specific components is used.

1. Misclassification component $\mathcal{L}_1$

$$\mathcal{L}_1(x, y, M_S) = \mathcal{L}(x, y, M_S)$$

Ensure misclassification

Attack methods (transferability)

Transferable Adversarial Perturbations (TAP)

A loss function with three specific components is used.

2. Feature separation component $\mathcal{L}_2$

Maximizing the distance between the features space representations of x and the feature space representation of x' for all layers of the source model.

Hypothesis (verified experimentally):
It allows craft adversarial perturbations that impact the inference process of the source model more strongly and deeply than simply considering misclassification, and improve the black-box transfer rate.

Attack methods (transferability)

QuantifiCare

Transferable Adversarial Perturbations (TAP)

A loss function with three specific components is used.

2. Feature separation component $\mathcal{L}_2$

$$\mathcal{L}_2(x, x') = \lambda \sum_{k=1}^{K} \|T(F_k(x)) - T(F_d(x'))\|_2 \qquad \lambda \in [0, 1]$$

$F_k(x)$: feature representation of an input x at layer d

$T(F_k(x)) = sign(F_k(x)) \odot (F_k(x))^a$: power normalization of $F_k(x)$ $\qquad \alpha \in [0, 1]$

Attack methods (transferability)

QuantifiCare

Transferable Adversarial Perturbations (TAP)

A loss function with three specific components is used.

2. High-frequency removing component $\mathcal{L}_3$

Remove high-frequency components that will be smoothed by convolution kernels, in the hope of having more transferable perturbations.

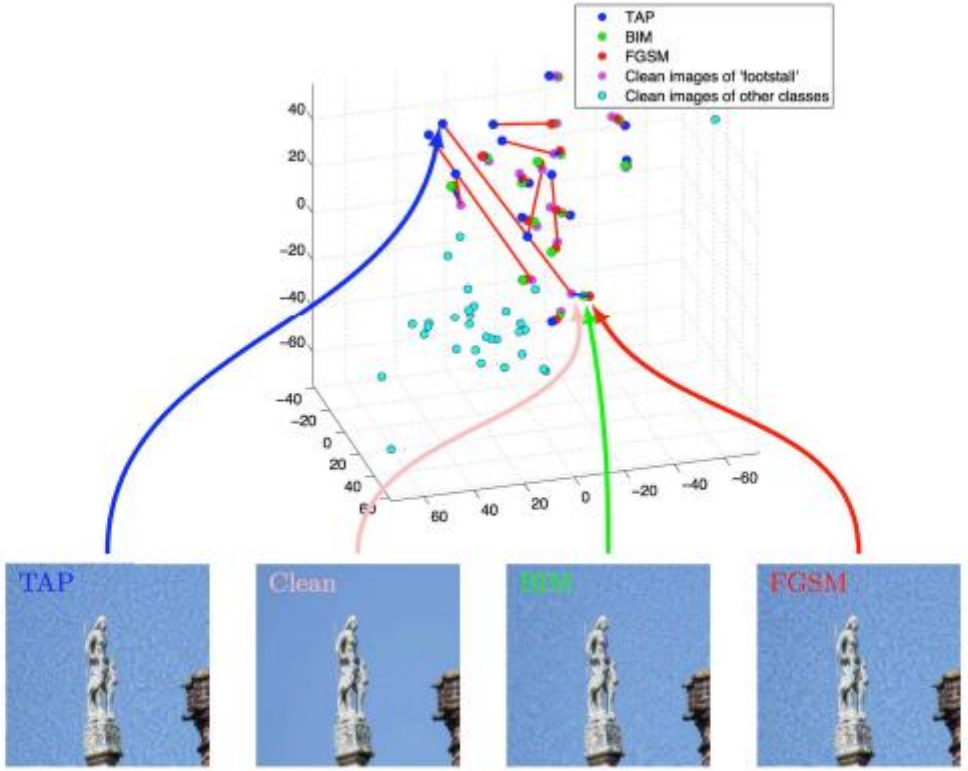
$$\mathcal{L}_3(x, x') = \eta \sum_i |R_i(x' - x, w)| \qquad \eta \in [0, 1]$$

$R_i(x' - x, w)$: i^{th} value of the response of a convolution between $x' - x$ and the kernel w, a box linear kernel (i.e. a kernel so that in the response each pixel is the average value of the neighbouring pixels)

Attack methods (transferability)

Transferable Adversarial Perturbations (TAP)

t-SNE visualization:
The TAP method allows to craft adversarial examples with larger perturbations in the intermediate layers than the other methods considered.



Attack methods (transferability)

Skip Gradient Method (SGM)

Wu et al. [144] propose SGM (Skip Gradient Method), a method to enhance the transferability of adversarial examples crafted on a source model with residual blocks (a combination of a residual module and a skip connection)

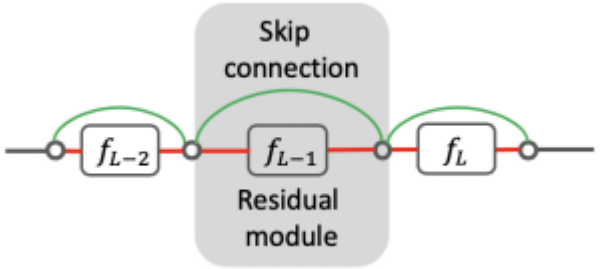
Idea:
Backpropagating using more skip connections than residual modules when crafting the adversarial example on the source model allows to rely more on low level information, which would be more transferable across different model architectures.

skip connections pass directly the information from one layer to another passing through the residual model

Attack methods (transferability)

Skip Gradient Method (SGM)

$z_i = z_{i-1} + f(z_{i-1})$ denotes the output of a residual block
z_{i-1}: skip connection part
$f(z_{i-1})$: residual module part



=> The gradient of z_i with respect to z_{i-1} equals $1 + \frac{\partial f}{\partial z_{i-1}}$

Toy example (gradient of the loss with respect to an input):

$$\frac{\partial \mathcal{L}}{\partial z_0} = \frac{\partial \mathcal{L}}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial z_0} = \frac{\partial \mathcal{L}}{\partial z_3}\left(1 + \frac{\partial f_3}{z_2}\right)\left(1 + \frac{\partial f_2}{z_1}\right)\left(1 + \frac{\partial f_1}{z_0}\right)$$

Attack methods (transferability)

Skip Gradient Method (SGM)

Toy example (gradient of the loss with respect to an input):

$$\frac{\partial \mathcal{L}}{\partial z_0} = \frac{\partial \mathcal{L}}{\partial z_3}\frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial z_0} = \frac{\partial \mathcal{L}}{\partial z_3}\left(1+\frac{\partial f_3}{z_2}\right)\left(1+\frac{\partial f_2}{z_1}\right)\left(1+\frac{\partial f_1}{z_0}\right)$$

(more generally) =>

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial z_L}\sum_{i=l}^{L-1}\left(\frac{\partial f_{i+1}}{\partial z_i}+1\right)\frac{\partial z_l}{\partial x}$$

=>

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial z_L}\sum_{i=0}^{L-1}\left(\frac{\partial f_{i+1}}{\partial z_i}+1\right)\frac{\partial z_0}{\partial x}$$

Attack methods (transferability)

Skip Gradient Method (SGM)

Finally:

To use more gradients form the skip connection, introduce a parameter γ in the gradient computation:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial z_L} \sum_{i=0}^{L-1} \left(\gamma \frac{\partial f_{i+1}}{\partial z_i} + 1 \right) \frac{\partial z_0}{\partial x}$$

Example (use in BIM or PGD)

Attack methods (transferability)

Skip Gradient Method (SGM)

Example:

BIM or PGD:

$$\delta^t = \alpha \operatorname{sign}(\nabla_x \mathcal{L}(x^{t-1}, y, M_\theta))$$
$$x^t = \operatorname{proj}_{B_p(x,\epsilon)}(x^{t-1} + \delta^t)$$
$$x^t = \operatorname{Clip}(x^t, \mathcal{Q})$$

BIM or PGD with SGM:

$$\delta^t = \alpha \operatorname{sign}(\frac{\partial \mathcal{L}}{\partial z_L} \sum_{i=0}^{L-1} \left(\gamma \frac{\partial f_{i+1}}{\partial z_i} + 1\right) \frac{\partial z_0}{\partial x})$$
$$x^t = \operatorname{proj}_{B_p(x,\epsilon)}(x^{t-1} + \delta^t)$$
$$x^t = \operatorname{Clip}(x^t, \mathcal{Q})$$

Defense methods

How to counter adversarial examples

Defense methods

Two types of defense methods:

1. Proactive schemes

Making the model more **robust in itself** against adversarial examples

Can be:
- ~~*provable: no adversarial example can be found at a certain distance of a clean example, if some condition is satisfied*~~
- empirical: no certification provided

Defense methods

Two types of defense methods:

1. Reactive schemes

Do not make the model more robust in itself against adversarial examples, act as **prevention**

Can be:
- detection schemes: recognizing adversarial examples, in order to treat them differently (e.g. discarding them, banning a user, etc.)
- purification: denoising an adversarial example, so that the model outputs the correct label on the denoised example

Defense methods

Reactive schemes

Major advantages:

1) Easily implemented:

No need to retrain the target model, as for empirical proactive schemes and some provable proactive schemes.

2) Scalability:

Scales better to any type of architecture than some provable proactive schemes.

Defense methods

Reactive schemes

The vast majority of proposed reactive schemes have been show to be bypassable.

Two main reasons:

- the setting considered for the evaluation of these defenses. It often refers as a gray-box setting (the adversary is not aware of the detection/purification mechanism). Under this hypothesis, the defense is shown to be efficient. However, simply relieving this hypothesis, as it is a good practice with adaptive adversaries, makes the defense useless.

- detection schemes tend to overfit to the type of adversarial examples they are designed for.

=> Increasingly less proposed in the defense literature

Defense methods

Proactive (empirical) schemes

Empirical proactive defense methods encompass all the defense schemes which do not provide a formal guarantee for the robustness of a model for some input.

=> no certified radius around a clean example is given, and thus the robustness of the model for this example can not be proven.

Good evaluation needed

Many works have emphasized this need, in order to foster a faster advance in the state of the art. Notably, some works have analyzed defense schemes proposed in scientific conferences to demonstrate some recurring flaws that lead to an over-evaluation of their efficiency.

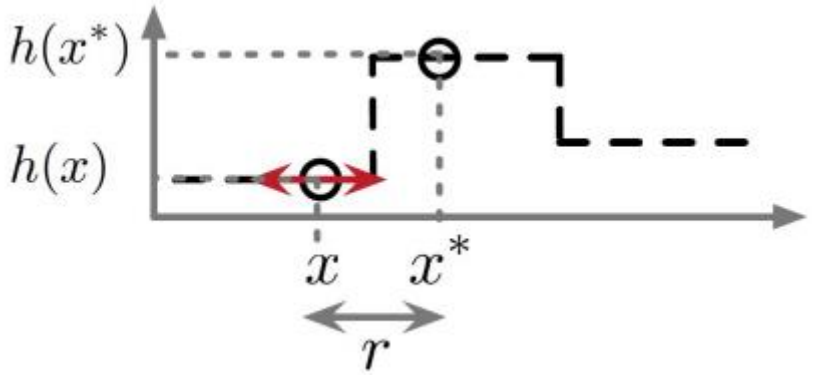
Defense methods

Proactive (empirical) schemes

Gradient masking (Athalye et al. [78]). False sense of security

Phenomenon where gradients of the model (loss, score function) with respect to the inputs are obfuscated.

=> Attack methods which rely on these gradients (gradient-based attacks) could therefore not find an adversarial example at some distance of a clean example, even if this adversarial example exists.



Defense methods

Proactive (empirical) schemes

Gradient masking can occur in three ways:

1. Shattered gradients: the defense scheme uses components that are either non-differentiable, introduce numeric instability or make the gradients non-existent or incorrect.
   In the case of differentiable gradients, following these gradients does not indicate the right direction to find adversarial examples.

2. Exploding or vanishing gradients.

Defense methods

Proactive (empirical) schemes

Gradient masking can occur in three ways:

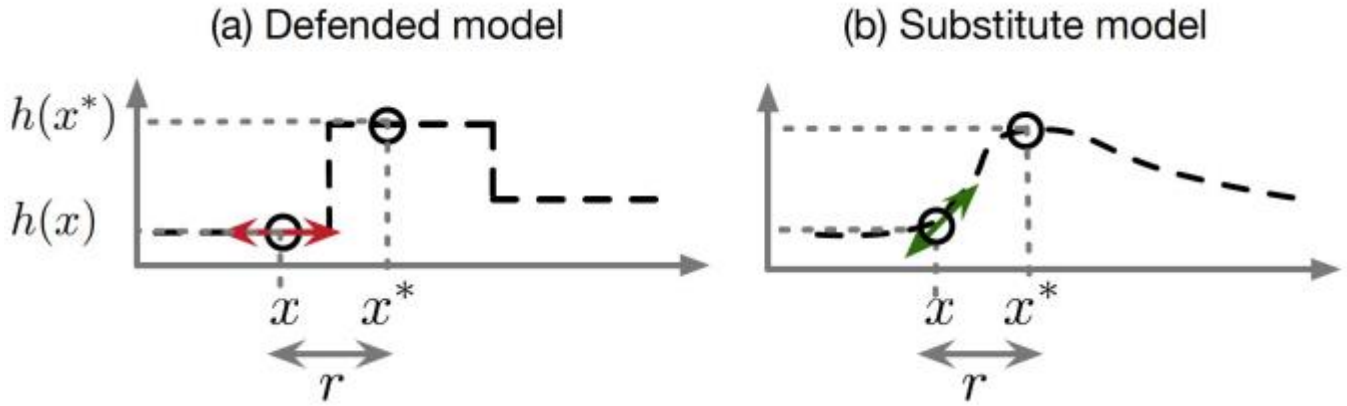
3. Stochastic gradients: the defense scheme uses randomization, which makes the gradient being not the same at each computation.
Using only one gradient computation can therefore be not sufficient to estimate the right direction to find adversarial examples.

Defense methods

Proactive (empirical) schemes

How to account for gradient masking ?

Shattered, exploding or vanishing gradients: not attack directly the target with gradient-based attacks, but rather use decision-based or score-based attacks, or leverage the transferability phenomenon.
This can allow to circusmvent existing gradient instability of the target model, as the gradient of the target model is not directly used.



Defense methods

Proactive (empirical) schemes

How to account for gradient masking ?

Shattered gradients: use BPDA (Athalye et al. [78]), i.e. approximate a non-differentiable operation with a differentiable approximation of it.

Stochastic gradients: use EOT (Expectation over Transformation, Athalye et al. [78]) method: instead of relying on one computation of the gradient, one may average many values of it, to reduce the impact of the randomization component included in the defense scheme.

Defense methods

Gradient regularization

Ross et al. [156] propose an intuitive method to defend a model against adversarial examples.

Idea:
Constraint the variations of the usual loss function $\mathcal{L}$ (e.g. $\mathcal{L}_{CE}$) with respect to variations of the input.

In practice:
Consider a loss function minimizing the gradients of the loss with respect to inputs:

$$\mathcal{L}_{GREG}(x, y, M) = \mathcal{L}_{CE}(x, y, M) + \lambda \left\| \nabla_x \mathcal{L}_{CE}(x, y, M) \right\|_2^2$$

Defense methods

Lipshitz constant

Cisse et al. [157] propose a defense method that aims at minimizing the phenomenon of an adversarial perturbation that is amplified as it goes through the model layers.

Idea:
Minimize the Lipschitz constant $Lip(M)$ of the target model M

Reminder:

Considering two metric spaces (X, d_x) and (Y, d_y), a function $f : X \to Y$ is said lipschitz continuous if there exists $K > 0$, such that $\forall (x_1, x_2) \in X \times X$:

$$d_y(f(x_1), f(x_2)) \leq K \; d_x(x_1, x_2)$$

K is said to be a lipschitz constant of f.

Defense methods

Lipshitz constant

Idea:
Minimize the Lipschitz constant $Lip(M)$ of the target model M

The Lipschitz constant of a model is the product of the Lipschitz constants of all layers.

Example:
Considering the l_2 norm, for a linear layer $W^t x$, the Lipschitz constant is the greatest singular value of the matrix of weight parameters W

=> Make W orthogonal

Defense methods

Lipshitz constant

Example:
Considering the l_2 norm, for a linear layer $W^t x$, the Lipschitz constant is the greatest singular value of the matrix of weight parameters W

=> Make W orthogonal

For each layer k, minimize the following term:

$$\frac{\beta}{2} \left\| W_k^T W_k - I \right\|_2^2$$

Optimizing it is expensive and may end in parameter far away from those resulting of the optimization of the classification loss !

Defense methods

Lipshitz constant

For each layer k, minimize the following term:

$$R_\beta(W_k) = \frac{\beta}{2} \left\| W_k^T W_k - I \right\|_2^2$$

Optimizing it is expensive and may end in parameter far away from those resulting of the optimization of the classification loss !

In practice:
Perform only one gradient step after the gradient step of the optimization relative to the classification loss

$$\nabla_{W_k} R_\beta(W_k) = \beta(W_k W_k^T - I)W_k$$

Defense methods

Lipshitz constant

In practice:
Perform only one gradient step after the gradient step of the optimization relative to the classification loss

$$\nabla_{W_k} R_\beta(W_k) = \beta(W_k W_k^T - I)W_k$$

Training procedure:

1) Gradient step for the classification loss:

$$W_k \leftarrow W_k - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(x_i, y_i, M_\theta) \qquad B\text{: batch size}$$

2) Gradient step for the orthogonality:

$$W_k \leftarrow (1+\beta)W_k - \beta W_k^T W_k W_k^T$$

Defense methods

Adversarial Training

Madry et al. [75] propose Adversarial Training, an efficient empirical defense scheme which has led to many variants of it.

Defense schemes consisting in improvements of Adversarial Training are currently the ones providing the best empirical results on various data sets (e.g. [158]).

Idea:
Account during training for the worst possible adversarial perturbation relatively to the threat model considered.

Defense methods

Adversarial Training

Standard training procedure:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L}(x, y, M_\theta)\right]$$

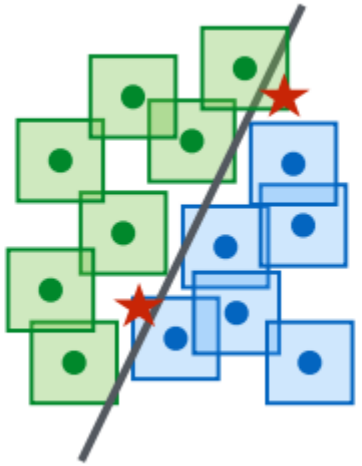
Adversarial (robust) training procedure:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_\theta)\right]$$

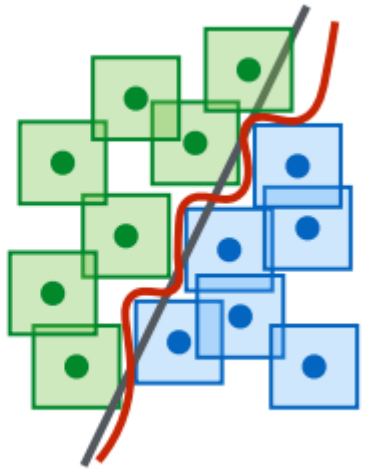
Defense methods

Adversarial Training

Illustration:



Standard

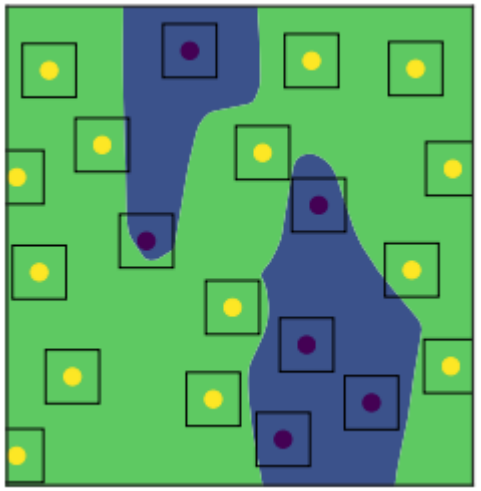


Robust (l_∞)

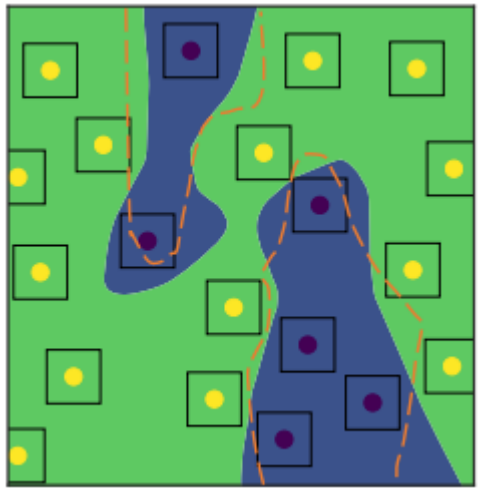
Defense methods

Adversarial Training

Illustration:



Standard



Robust (l_∞)

Defense methods

Adversarial Training

Adversarial (robust) training procedure:

outer

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_\theta) \right]$$

inner

In practice during training:

- **i)** For each batch, for each example-label pair (x, y), perform the l_∞ PGD attack for a fixed number of steps k to get x'
- **ii)** Train the model on (x', y) (train the model to predict the label y for example x')

Defense methods

Adversarial Training

For N epochs:

For each batch:

For each input-label pair (x_i, y_i):

Perform the l_∞ PGD attack for k iterations

Starting point: $x_i^0 = x_i + \eta \qquad \eta \sim \mathcal{U}(-\epsilon, \epsilon)$

Repeat k times:

$$\delta^t = \alpha \operatorname{sign}(\nabla_{x_i} \mathcal{L}(x_i^{t-1}, y, M_\theta))$$

$$x_i^t = \operatorname{proj}_{B_p(x_i, \epsilon)}(x_i^{t-1} + \delta^t)$$

$$x_i^t = \operatorname{Clip}(x_i^t, \mathcal{Q})$$

$$x_i' = x_i^k$$

Calculate gradients and update the loss function: $\theta \leftarrow \theta - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta \mathcal{L}(x_i', y_i, M_\theta)$

Defense methods

Adversarial Training for Free

A big flaw of Adversarial Training:

The robust generalization gap

The robustness on train examples is way much higher than on test examples.

Defense methods

Adversarial Training for Free

Adversarial training: For each example encountered during training, need to perform k gradient calculations !

Shahafi et al. [159] propose an alternative to alleviate this burden.

Idea:

Recycle the gradient of the attack when updating model parameters.

Update both model parameters and adversarial perturbation in one pass rather than separately.

Defense methods

Adversarial Training for Free

Idea:

Recycle the gradient of the attack when updating model parameters.

Update both model parameters and adversarial perturbation in one pass rather than separately.

Adversarial training :
k gradient steps for the inner (maximization) problem, then one gradient step for the outer (minimization problem)

Free Adversarial training :
1) Both outer and inner problem calculated simultaneously

Defense methods

Adversarial Training for Free

Adversarial training :
k gradient steps for the inner (maximization) problem, then one gradient step for the outer (minimization problem)

Free Adversarial training :
Both gradient step for both outer and inner problem calculated simultaneously

=> Only one adversarial perturbation per example (like FGSM)

=> Repeat m times per batch to account for multiple adversarial perturbation updates

=> Perform only $\frac{N}{m}$ epochs

Defense methods

Adversarial Training for Free

Adversarial training :

1) k gradient steps for the inner (maximization) problem, then one gradient step for the outer (minimization problem)

Algorithm 1 PGD adversarial training for T epochs, given some radius ϵ, adversarial step size α and N PGD steps and a dataset of size M for a network f_θ

for $t = 1 \dots T$ **do**
  for $i = 1 \dots M$ **do**
    // Perform PGD adversarial attack
    $\delta = 0$ *// or randomly initialized*
    for $j = 1 \dots N$ **do**
      $\delta = \delta + \alpha \cdot \text{sign}(\nabla_\delta \ell(f_\theta(x_i + \delta), y_i))$
      $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
    end for
    $\theta = \theta - \nabla_\theta \ell(f_\theta(x_i + \delta), y_i)$ *// Update model weights with some optimizer, e.g. SGD*
  end for
end for

Defense methods

Adversarial Training for Free

Free Adversarial training :

Both gradient step for both outer and inner problem calculated simultaneously

Algorithm 2 "Free" adversarial training for T epochs, given some radius ϵ, N minibatch replays, and a dataset of size M for a network f_θ

$\delta = 0$
// Iterate T/N times to account for minibatch replays and run for T total epochs
for $t = 1 \dots T/N$ **do**
  for $i = 1 \dots M$ **do**
    // Perform simultaneous FGSM adversarial attack and model weight updates T times
    for $j = 1 \dots N$ **do**
      // Compute gradients for perturbation and model weights simultaneously
      $\nabla_\delta, \nabla_\theta = \nabla \ell(f_\theta(x_i + \delta), y_i)$
      $\delta = \delta + \epsilon \cdot \text{sign}(\nabla_\delta)$
      $\delta = \max(\min(\delta, \epsilon), -\epsilon)$
      $\theta = \theta - \nabla_\theta$ *// Update model weights with some optimizer, e.g. SGD*
    end for
  end for
end for

Defense methods

Adversarial Training for Free

Illustration:

Training	Evaluated Against					Train Time (min)
	Nat. Images	PGD-20	PGD-100	CW-100	10 restart PGD-20	
Natural	**95.01%**	0.00%	0.00%	0.00%	0.00%	**780**
Free $m = 2$	91.45%	33.92%	33.20%	34.57%	33.41%	816
Free $m = 4$	87.83%	41.15%	40.35%	41.96%	40.73%	800
Free $m = 8$	85.96%	**46.82%**	**46.19%**	**46.60%**	**46.33%**	785
Free $m = 10$	83.94%	46.31%	45.79%	45.86%	45.94%	785
7-PGD trained	87.25%	45.84%	45.29%	46.52%	45.53%	5418

Defense methods

Multi-Steepest Descent (MSD) for Adversarial Training

Maini et al. [162] propose an Adversarial Training method which allows to train models which are robust to l_∞, l_2 and l_1 adversarial examples.

Idea:

Perform a single step of the PGD attack for each $p \in \{1, 2, \infty\}$, keeping the adversarial perturbation resulting in the biggest loss value as the initial adversarial perturbation for the next step.

Algorithm 1 Multi steepest descent for learning classifiers that are simultaneously robust to ℓ_p attacks for $p \in \mathcal{S}$

Input: classifier f_θ, data x, labels y
Parameters: ϵ_p, α_p for $p \in \mathcal{S}$, maximum iterations T, loss function ℓ
$\delta^{(0)} = 0$
for $t = 0 \dots T-1$ **do**
  for $p \in \mathcal{S}$ **do**
    $\delta_p^{(t+1)} = P_{\Delta_{p,\epsilon}}(\delta^{(t)} + v_p(\delta^{(t)}))$
  end for
  $\delta^{(t+1)} = \arg\max_{\delta_p^{(t+1)}} \ell(f_\theta(x + \delta_p^{(t+1)}), y)$
end for
return $\delta^{(T)}$

Defense methods

Misclassification Aware adveRsarial Training (MART)

Wang et al. [163] propose a defense scheme based on the fact of treating differently correctly classified and misclassified examples during Adversarial Training.

Motivation:

Experiments show different behaviors between correctly (S^+) and incorrectly (S^-) classified examples:

1) Not perturbing misclassified examples during the training leads to decreased final robustness

2) Using a weak perturbation (FGSM instead of PGD) leads to decreased robustness when using a weak perturbation for S^- and to no robustness change when using a weak perturbation for S^+ only

Defense methods

Geometry-Aware Instance-Reweighted Adversarial Training (GAIRAT)

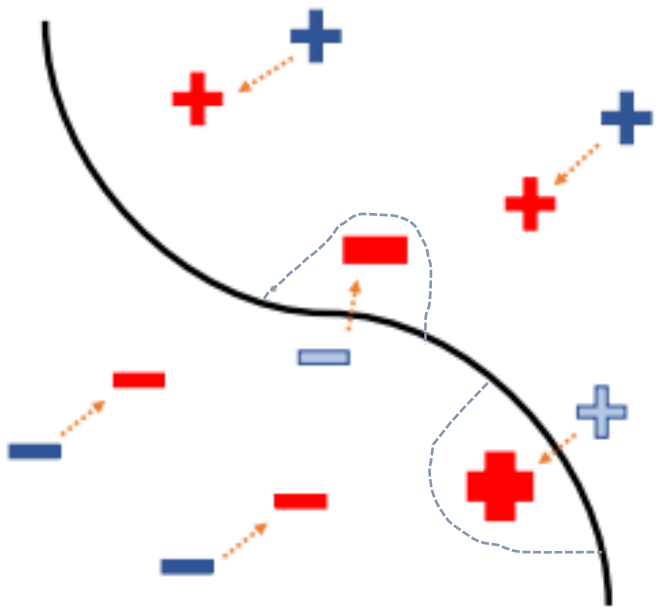
Zhang et al. [165] propose an enhanced version of Adversarial Training, that treats differently examples whether they are close or far from the decision boundary.

Idea:
Adversarial examples crafted from examples close to the decision boundary should be treated with more importance than those crafted from examples further from the decision boundary, as they are more responsible for the final shape of a robust decision boundary.

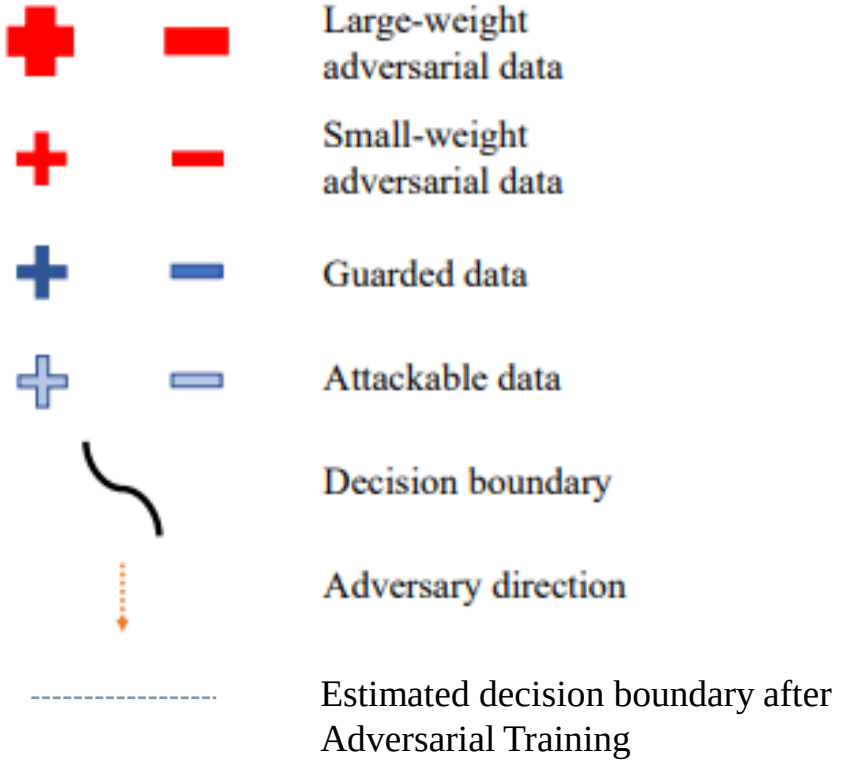
Defense methods

Geometry-Aware Instance-Reweighted Adversarial Training (GAIRAT)

Illustration:



Larg red: large weights
Small red: small weights



Defense methods

Geometry-Aware Instance-Reweighted Adversarial Training (GAIRAT)

Standard loss for Adversarial Training:

$$\frac{1}{n}\sum_{i=1}^{B}\mathcal{L}(x', y', M_\theta)$$

GAIRAT loss:

$$\frac{1}{n}\sum_{i=1}^{B} w(x_i, y_i)\,\mathcal{L}(x_i', y_i', M_\theta)$$

where the closer x_i to the decision boundary, the bigger $w(x_i, y_i)$

=> During training, the more the decision boundary is made far from a clean example, the smaller the weights assigned to the corresponding adversarial example

Defense methods

Geometry-Aware Instance-Reweighted Adversarial Training (GAIRAT)

How is computed $w(x_i, y_i)$?

$w(x_i, y_i)$ is based on $\kappa(x_i, y_i)$, the number K of PGD iterations with step size α needed to craft a misclassified adversarial example x' from x

The distance to the decision boundary is thus approximated with $\alpha\, K$

Example:

$$w(x_i, y_i) = 1 - \frac{\kappa(x_i, y_i)}{K+1}$$

Defense methods

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Wu et al. [222] investigate the link between the robust generalization gap of Adversarial Training (the generalization gap on adversarial examples) and the weight loss landscape (the landscape of the loss with respect to weight perturbations).

The lower the robust generalization gap
(small robustness difference between train and test examples), the better.

Defense methods

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Adversarial Training:

$$\min_{\theta} \rho(\theta) = \min_{\theta} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_\theta) \right]$$

Experiments:

Visualize weight loss landscape $\rho(\theta + \alpha\, d)$ for different α values where d is a random direction

It is looked at the Adversarial Training loss $\rho(\theta)$ when weight parameters d are moved along a direction with different intensities

Defense methods

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Experiments:

Visualize weight loss landscape $\rho(\theta + \alpha\, d)$ for different α values where d is a random direction

It is looked at the Adversarial Training loss $\rho(\theta)$ when weight parameters d are moved along a direction with different intensities

Results:

Models with small robust generalization gaps correlate with flat weigth loss landscapes

Defense methods

QuantifiCare

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Idea:

Perform Adversarial Training while enforcing a flat weight loss landscape
(in others words, ensuring robustness to weight perturbation)

1) Adversarial Training, which induces a flat input loss landscape (the landscape of the loss with respect to input perturbations), generates adversarial perturbations which only affect one example at a time.

2) Weight perturbations could produce a more global effect by affecting multiple examples.
=> AWP could reduce the robust generalization gap

Defense methods

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Adversarial Training:

$$\min_{\theta} \rho(\theta) = \min_{\theta} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_\theta) \right]$$

Adversarial Training with Weight Perturbation:

$$\min_{\theta} \max_{v \in \mathcal{V}} \rho(\theta + v) = \min_{\theta} \max_{v \in \mathcal{V}} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_{\theta+v}) \right]$$

Adversarial Training with Weight Perturbation: $\rho(\theta + v) = \rho(\theta) + (\rho(\theta + v) - \rho(\theta))$

=> Adversarial Training (first term) and flatness of the weight loss landscape(second term)

Defense methods

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Adversarial Training with Weight Perturbation:

$$\min_{\theta} \max_{v \in \mathcal{V}} \rho(\theta + v) = \min_{\theta} \max_{v \in \mathcal{V}} \left[\max_{x' \in B_\infty(x, \epsilon)} \mathcal{L}(x', y, M_{\theta+v}) \right]$$

$\mathcal{V}$: adversary capacity for the weight perturbation

$$\|v_l\| \leq \gamma \|w_l\|$$

where:
v_l denotes the weight perturbation for layer l
w_l denotes weights for layer l

Defense methods

Adversarial Training with Adversarial Weight Perturbation (AT-AWP)

Adversarial Training with Weight Perturbation:

$$\min_{\theta} \max_{v \in \mathcal{V}} \rho(\theta + v) = \min_{\theta} \max_{v \in \mathcal{V}} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_{\theta+v}) \right]$$

Optimization:

$\max_{x' \in B_\infty(x,\epsilon)}$ (Adversarial Training): projected l_∞ PGD on $\mathcal{L}(x', y, M_{\theta+v})$ with respect to x' (on $B_\infty(x, \epsilon)$)

$\max_{v \in \mathcal{V}} \rho(\theta + v)$ (Weight Perturbation): projected l_2 PGD on $\mathcal{L}(x', y, M_{\theta+v})$ with respect to v (on $\mathcal{V}$)

Defense methods

Adversarial Feature Stacking

Liu et al. [168] propose AFS (Adversarial Feature Stacking), a defense method against adversarial examples designed to reduce the trade-off between accuracy and robustness in Adversarial Training.

Idea:
Consider many models trained with Adversarial Training with different adversarial budgets ($\|\epsilon\|_\infty$) to take advantage of features with various levels of usefulness and robustness.

Considering $\epsilon_1 > \epsilon_2$, models trained with AT with the ϵ_1 value will learn features which are more robust but less useful than models trained with AT with the ϵ_2 value. This results in the model trained with ϵ_1 showing a greater robust accuracy but a smaller clean accuracy than the model trained with ϵ_2

Defense methods

Adversarial Feature Stacking

Idea:
Consider many models trained with Adversarial Training with different adversarial budgets ($\|\epsilon\|_\infty$) to take advantage of features with various levels of usefulness and robustness.

In practice:

Consider k models, each of them trained with AT with a different ϵ value.

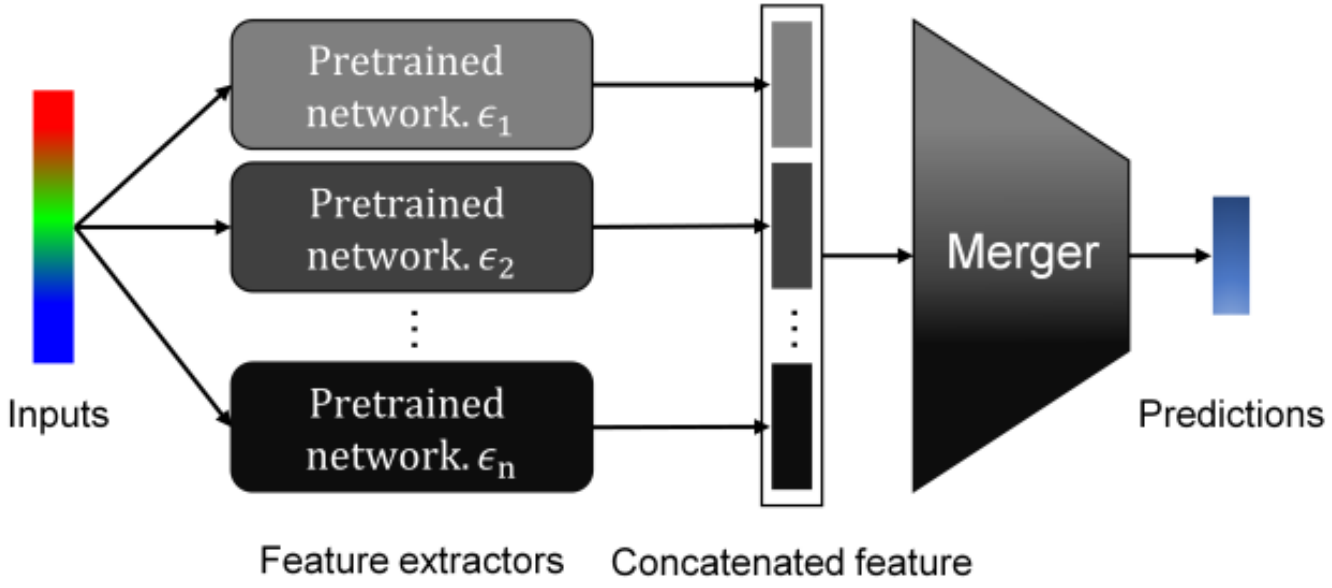
A merger model J takes as inputs $g_1(x), \ldots, g_n(x)$ and outputs a confidence score vector.

$g_i(x)$: features (output of the layer before the logits layer) of the i^{th} model

Defense methods

Adversarial Feature Stacking

Scheme:



Defense methods

Adversarial Feature Stacking

Training the merger model J :

Consider the objective:

$$\min_{\theta} \quad \alpha \mathcal{L}(x, y, J_\theta) + (1-\alpha)\left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(x', y, M_\theta)\right]$$

α allows to calibrate between accuracy (first term) and robustness (second term)

The k models trained with Adversarial Training are frozen.
Only the weights relative to the merger model J are modified.

Defense methods

Confidence-Calibrated Adversarial Training (CCAT)

Stutz et al. [170] propose to enchance Adversarial Training so that it is still robust outside the considered l_∞ ball and for other threat models (e.g. l_2)

Idea:
Perform classical Adversarial Training while training the model to output a “uniform distribution” for adversarial examples encountered during training

The more the examples at the edge of the l_∞ ball, the more uniform the distribution

=> Make the model generalize to bigger perturbations than those seen during training

=> Output a uniform distribution for adversarial examples crafted considering other threat models, making all type of adversarial examples detectable afterwards.

Defense methods

Confidence-Calibrated Adversarial Training (CCAT)

In practice:

1) Craft an adversarial example with respect to some l_∞ threat model (with budget ϵ)

$$\delta = \max_{\delta \leq \|\epsilon\|_\infty} \max_{k \neq y} F_k(x+\delta, y) \qquad x' = x + \delta$$

2) Build the "uniform distribution" that depends on δ

$$\lambda(\delta) = \left(1 - min\left(1, \frac{\|\delta\|_\infty}{\epsilon}\right)\right)^\rho$$

$$\begin{cases} \text{if } \|\delta\|_\infty \geq \epsilon,\ \lambda(\delta) = 0 \\ \text{decreases as } \|\delta\|_\infty \text{ increases} \end{cases}$$

$$\hat{y} = \lambda(\delta) one\_hot(y) + (1 - \lambda(\delta))\frac{1}{K} \qquad K \text{ : number of classes}$$

Defense methods

Confidence-Calibrated Adversarial Training (CCAT)

In practice:

- Model training, for a batch of size B:

$$\min_{\theta} \quad \frac{1}{B/2} \sum_{i=1}^{B/2} \mathcal{L}(x_i, y_i, M_\theta) + \frac{1}{B/2} \sum_{i=B/2}^{B} \mathcal{L}(x'_i, \hat{y}_i, M_\theta)$$

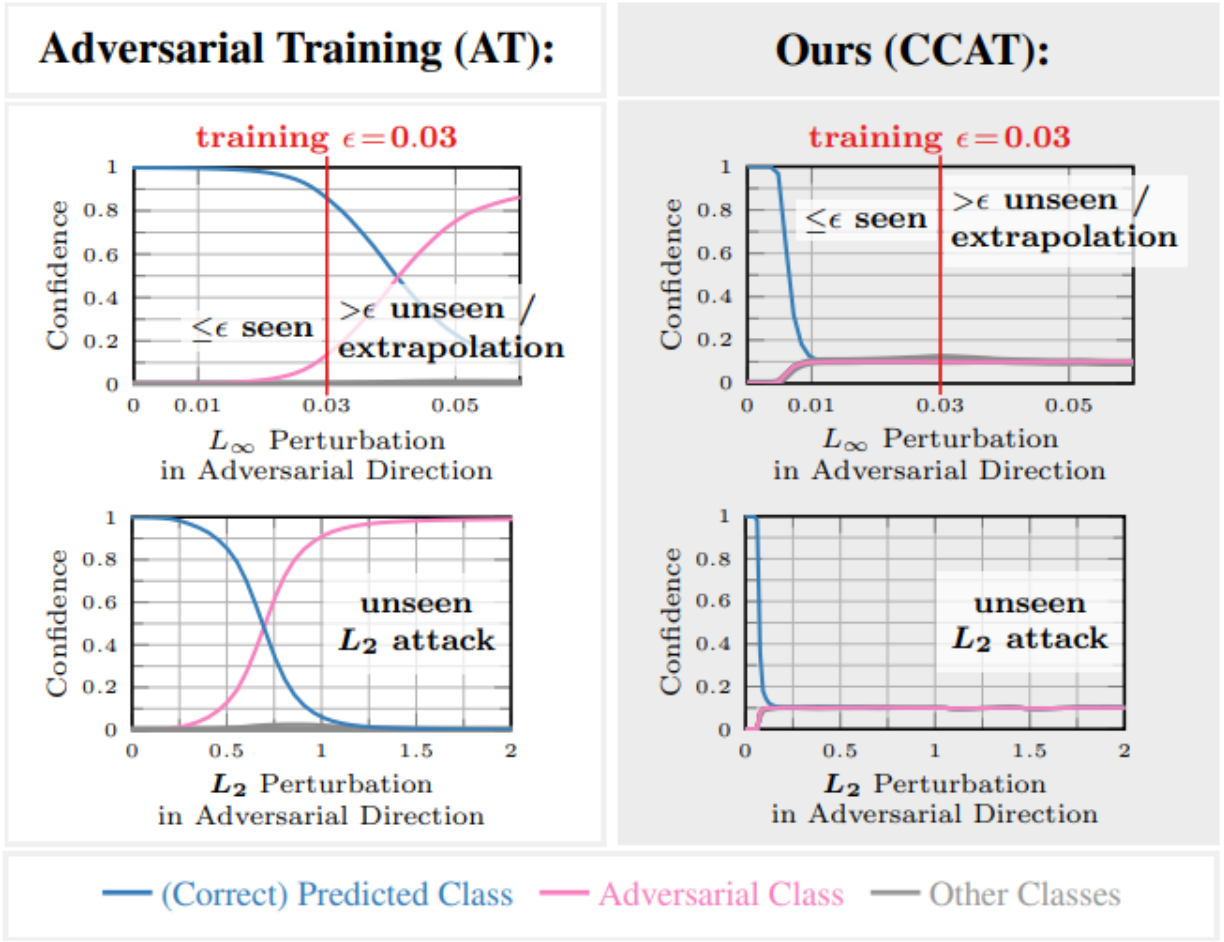
- Prediction (with detection scheme):

Threshold with the biggest confidence score

Defense methods

Confidence-Calibrated Adversarial Training (CCAT)

Illustration:



Defense methods

Adversarially Robust Distillation (ARD)

Goldbum et al. [167] propose a method to transfer robustness from a *teacher model* ($\mathcal{T}$) to a *student model* (S)

In practice:
Train the student model by minimizing the Kullback-Leibler divergence between its score vector and the one of the teacher model.

$$\min_{\theta} \quad KL(T(x), S_\theta(x))$$

$T(x)$: confidence score vector of the teacher model

$S_\theta(x)$: confidence score vector of the teacher model

Defense methods

Robust Local Features for Adversarial Training (RLFAT)

Song et al. [169] propose an enhancement to Adversarial Training which allows the trained model to rely more on local features.

Motivation:

1) Adversarially trained models rely more on global features than classically trained models.

2) Adversarially trained models generalize with difficulty for clean and adversarial examples (i.e there is a smaller test accuracy than for standard models, and there is a consequent gap between adversarial robustness on training and test set).

3) Standard models rely more on local features than adversarially trained models.

Defense methods

Robust Local Features for Adversarial Training (RLFAT)

Goal:

Perform Adversarial Training while accounting more for local features

=> higher accuracy on test set examples

=> smaller robust generalization gap

Idea:

During Adversarial Training, not considering the adversarial examples as they are

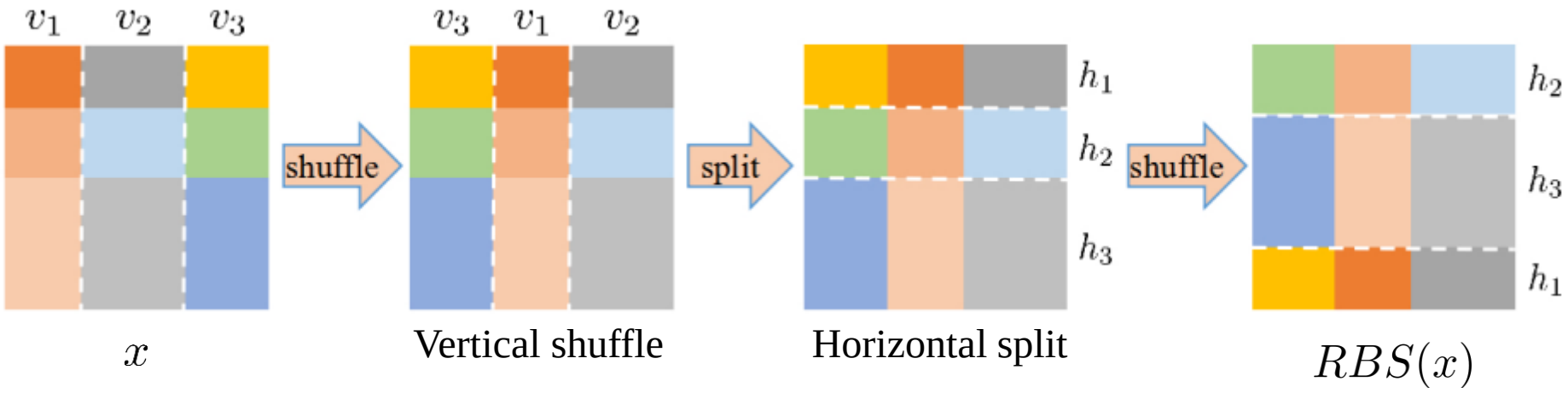
But split them in many blocks and shuffle these blocks to break the global structure of features

=> The model is forced to rely more on local features

Defense methods

Robust Local Features for Adversarial Training (RLFAT)

Random Block Shuffle (RBS):



Defense methods

Robust Local Features for Adversarial Training (RLFAT)

Model training:

$$\min_{\theta} \left[\max_{x' \in B_\infty(x,\epsilon)} \mathcal{L}(RBS(x'), y, M_\theta) \right] + \|g(RBS(x')) - g(x')\|_2^2$$

- First term: Adversarial training with RBS
- Second term: minimize the difference between features learned for adversarial examples passed through or not RBS.

Account between train and test differences (RBS is not applied at test time)

Defense methods

Jacobian Adversarially Regularized Networks (JAN)

Chan et al. [171] show that forcing a model to have salient Jacobian matrices for the loss induces better robustness against adversarial examples.

Reminder:

For an input space of dimension $d = n^2$ (e.g. images of size $n \times n$), the Jacobian matrix is given by:

$$J(x) = [\nabla_{x_i} \mathcal{L}_{CE}(x, y, M_\theta)]_{i \in [1,d]}$$

---

1) A salient Jacobian matrix outputs features that are interpretable by humans.

2) Robust models are known to have more salient features.

Goal:

Train models to have salient Jacobian matrices

Defense methods

Jacobian Adversarially Regularized Networks (JAN)

Goal:
Train models to have salient Jacobian matrices

In practice:
Leverage an adversarial framework where a generator competes with a discriminator to have the jacobian matrice $J(x)$ the closest as possible to x

- Generator: $G_{\theta,\psi} = f_{apt}(J(x))$ with: f_{apt} a convolution layer to $[-1, 1]$ with parameters ψ
  $J(x)$ the Jacobian matrix of model M_θ
- Discriminator: f_{disc} to $[0, 1]$ with parameters ϕ

Defense methods

Jacobian Adversarially Regularized Networks (JAN)

Training procedure:

Generator and discriminator (adversarial loss):

$$\mathcal{L}_{ADV} = \mathbb{E}_x[log(f_{disc}(x))] + \mathbb{E}_x[1 - log(1 - f_{disc}(f_{apt}(J(x))))] \qquad \psi,\ \phi$$

Model loss :

$$\mathcal{L}_{tot} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{ADV} \qquad \theta$$

At each training iteration :

Model is trained by minimizing $\mathcal{L}_{tot}$ (parameters θ)

Generator and discriminator are trained (parameters $\psi,\ \phi$)

Defense methods

Bit Planes Feature Consistency (BPFC)

Addepalli et al. [172] propose to train a model that behaves similarly for an example and its high bit plane version (i.e. the example where only the most important bits are kept).

Motivation:
Human brains are not fooled by adversarial examples and rely on large magnitude components to make a decision (then slightly refines it by looking at details).

Goal:
Enforce a model to behave similarly on an example and its high bit plane version

=> it does not rely on low bit plane information where adversarial perturbation is located.

Defense methods

Bit Planes Feature Consistency (BPFC)

Goal:
Enforce a model to behave similarly on an example and its high bit plane version

=> it does not rely on low bit plane information where adversarial perturbation is located.

In practice:
Considering an example x on n bits (possible values from 0 to $2^n - 1$), is computed $q_k(x)$, its quantized version on $n - k + 1$ bits, where the k least important bits are eliminated.

- Loss function:

$$\mathcal{L}_{CE}(x, y, M_\theta) + \lambda \left\| h(x) - h(q_k(x)) \right\|_2^2$$

Any questions ?