Defenses against adversarial attacks

Bhiksha Raj, Joseph Keshet, Raphaël Olivier

Interspeech 2019

15 Sep 2019, Graz

Introduction

We are now interested in the target's point of view.

A **model** (e.g. a neural network) is performing a **task** (e.g. classification) with some level of success (e.g. 95% accuracy).

An **adversary** is attacking the model (e.g. with FGSM perturbations).

Objective : to perform the task (maintaining a similar accuracy) *under attack*.

Threat model

- Adversarial Knowledge-level
  - White-box: data, architecture, gradients
  - Black-box: only inputs and outputs
- Adversarial Specificity:
  - Targeted: maximize probability of targeted class
  - Nontargeted: minimize probability of correct class
- Adversarial Objective:
  - Maximize (mis)classification under fixed distortion rate (ϵ)
  - Minimize the distortion rate needed for classification

Defense desiderata (for now)

- Robustness
  - Performance under attack compares to performance without attacks
  - Effective under different attack schemes
- Efficiency:
  - Low impact on accuracy
  - Low impact on speed
  - Low impact on existing architecture

We will introduce more criteria gradually.

Outline

- A discussion on the origin/potential inevitability of adversarial vulnerability.
- An introduction of past and current adversarial defenses, by “family”, and (+/-) from most “naïve” to most “advanced”.
- Towards the end, an emphasis on audio-based methods.

Understanding adversarial examples

Understanding adversarial examples

Open question : propose models explaining the existence and the extent of adversarial examples in the framework of machine learning algorithms.

The answer could explain to what extent we can hope to defend against them.

We will go over some historical explanations and discuss their limits.

Low probability pockets

(Szegedy et al, 2013)

Early explanation : Models learn representations with “low-probability pockets” where adversarial examples lie. Those pockets come from the high non-linearity of neural network models.

This explanation does not hold well : adversarial examples exist as well for simpler models and seem data dependent (cf transferability).

High-dimension phenomenon

(Gilmer et al. 2018)

Adversarial examples can be linked to the *high dimension of the data* .

The authors use a synthetic task : classify in n-dimensional spheres.

They show that on this dataset, the *average distance of a point to the nearest error*:

$$d(E) = \mathbb{E}_{x \sim p}[d(x, E)]$$

can be upper bounded based on the measure of the error set E:

$$\mu(E) = \mathbb{P}_{x \sim p}[x \in E]$$

and that examples are vulnerable to perturbations of size $\mathrm{O}\left(\frac{1}{\sqrt{n}}\right)$.

High-dimension phenomenon

(Gilmer et al. 2018)

But…

The model does not necessarily generalize to any dataset (Ilyas et al. 2019)

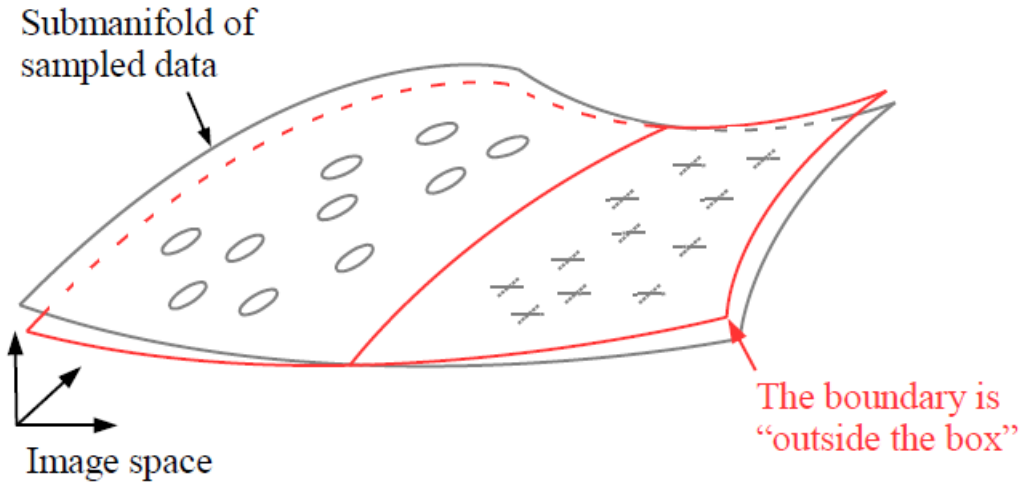
In particular it does not account for the existence of *some* defenses and robust classifiers on *some* datasets.

Therefore it is not fully satisfying.

Boundary tilting

(Tanay et al. 2016)

The classifier boundary isn't normal to the underlying data manifold, explaining that correct examples are close to wrongly classified ones.



That phenomenon can be explained by overfitting.

Boundary tilting

(Tanay et al. 2016)

The boundary tilting model explains adversarial attacks with classifiers not learning the correct data distribution, and suggests that the solution is in regularizing the models.

But…

There are reasons to believe that adversarial inputs are inherent to the data distribution.

Non-robust features

(Ilyas et al 2019)

When performing a task, models :

- learn features correlated to the instance label,
- that can be robust, i.e. they stay correlated when applying small perturbations to the input
- or they can be non-robust, i.e. they can become non-correlated or negatively correlated under small perturbations

Non-robust features

(Ilyas et al 2019)

Regular machine learning models tend to learn non-robust features, while models using a defense (such as adversarial training for some class of perturbation) learn only robust features.

An adversarial example has robust features still correlated with the true label (by definition) but non-robust features correlated with another label.

Non-robust features

(Ilyas et al 2019)

The authors propose a surprising experiment.

First they create a dataset where instances only have non-robust features correlated with their label (i.e. a dataset of adversarial examples, associated to the *fake* label).

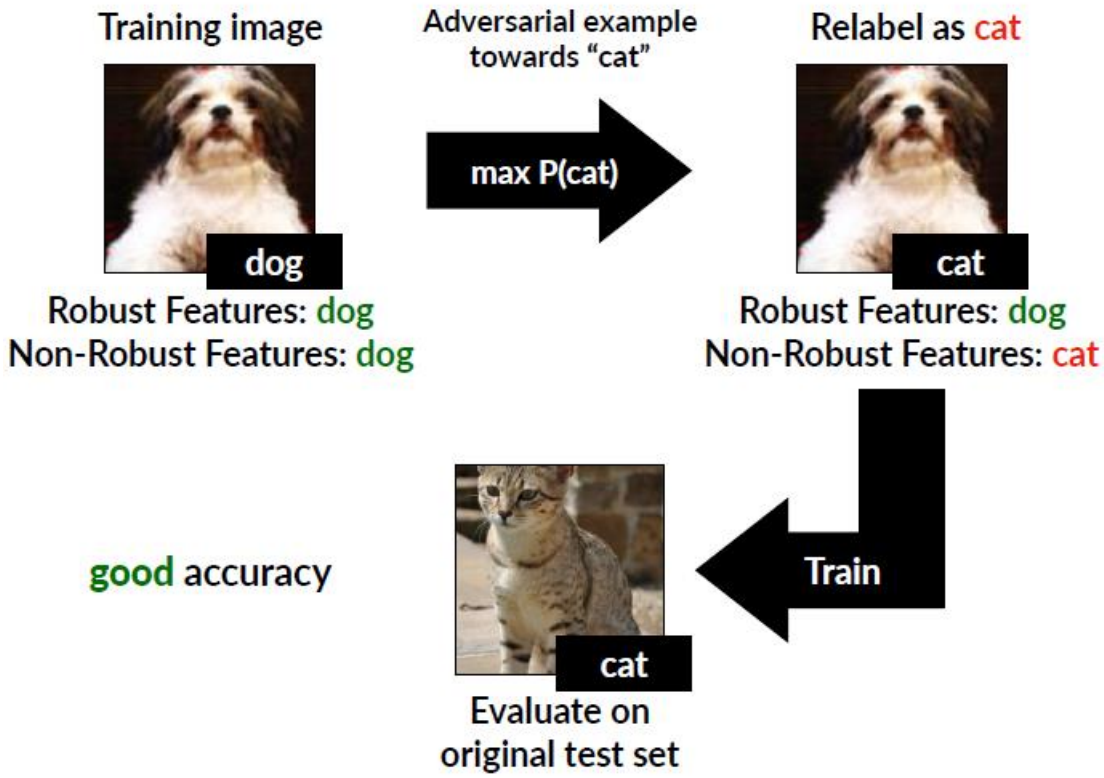
They do so by sampling a target instance (x, y) and a random instance (x_0, y_0), then applying gradient descent $x_0 \rightarrow \mathrm{x}_{\mathrm{new}}$ until $\mathrm{x}_{\mathrm{new}}$'s non robust representation converges to that of x. They then add (x_{new}, y) to the new dataset.

Non-robust features

(Ilyas et al 2019)

Then they train a standard model on that new dataset.

The model *learns predictive features* for the **original dataset**.



Non-robust features

(Ilyas et al 2019)

Source Dataset	Dataset	
	CIFAR-10	ImageNet_R
$\mathcal{D}$	95.3%	96.6%
$\widehat{\mathcal{D}}_{rand}$	63.3%	87.9%
$\widehat{\mathcal{D}}_{det}$	43.7%	64.4%

→ Adversarial examples share predictive features for the class they get predicted as, even if that class is “wrong” from the eye’s perspective.

Adversarial training

Adversarial training

(Goodfellow et al. 2014)

Definition : During training, have the model learn to classify correctly adversarial examples.

Aka “brute force” defense.

First introduced for linear models.

Adversarial training

(Goodfellow et al. 2014)

Logistic regression (standard setting) : minimize

$$\log\left(1+\exp\left(-y(w^T x+b)\right)\right)$$

through gradient descent.

Adversarial training

(Goodfellow et al. 2014)

Adversarial setting : minimize instead that loss on a worst case ϵ-perturbation on x :

$$\log\left(1+\exp\left(-y\left(w^T\left(x-\epsilon sign(w)\right)+b\right)\right)\right)$$

$$=\log\left(1+\exp\left(y\left(\epsilon\|w\|_1-w^Tx-b\right)\right)\right)$$

Adversarial training

(Goodfellow et al. 2014)

Adversarial setting : minimize instead that loss on a worst case ϵ-perturbation on x :

$$\log\left(1+\exp\left(-y\left(w^T\left(x-\epsilon sign(w)\right)+b\right)\right)\right)$$

$$=\log\left(1+\exp\left(y\left(\epsilon\|w\|_1-w^Tx-b\right)\right)\right)$$

Note : this differs from L1 regularization :

$$\log\left(1+\exp\left(-y\left(w^Tx+b\right)\right)\right)+\epsilon\|w\|_1$$

Adversarial training

(Goodfellow et al. 2014)

Extension to deep networks : for any training objective $J(\theta, x, y)$, use the adversarial objective $J(\theta, x + \epsilon\, sign(\nabla_x J(\theta, x, y)), y)$.

One can interpolate the standard and adversarial objectives in gradient descent.

Alternatively, one can first generate adversarial examples, add them to the training data, and train normally. (Kurakin et al. 2016)

Gradient regularization

(Lyu et al. 2015)

A more general framework.

Minimize over θ

$$\max_{\|\eta\|_p \leq \epsilon} J(\theta, x + \eta, y)$$

With Taylor approximation : $\max_{\eta}(J(\theta, x, y) + \nabla_x J . \eta)$ s.t. $\|\eta\|_{p \leq \epsilon}$

Gradient regularization

(Lyu et al. 2015)

Using a Lagrangian multiplier we obtain :

$$\eta = \epsilon sign(\nabla J) \left(\frac{|\nabla J|}{\|\nabla J\|_{\frac{p}{p-1}}} \right)^{\frac{1}{p-1}}$$

Applying $p = \infty$ we get the previous value of the perturbation $\epsilon sign(\nabla J)$

Strengths of adversarial training

- It is a **simple** method to use and implement
- It fits into **general** frameworks
- It can be adapted to many attacks

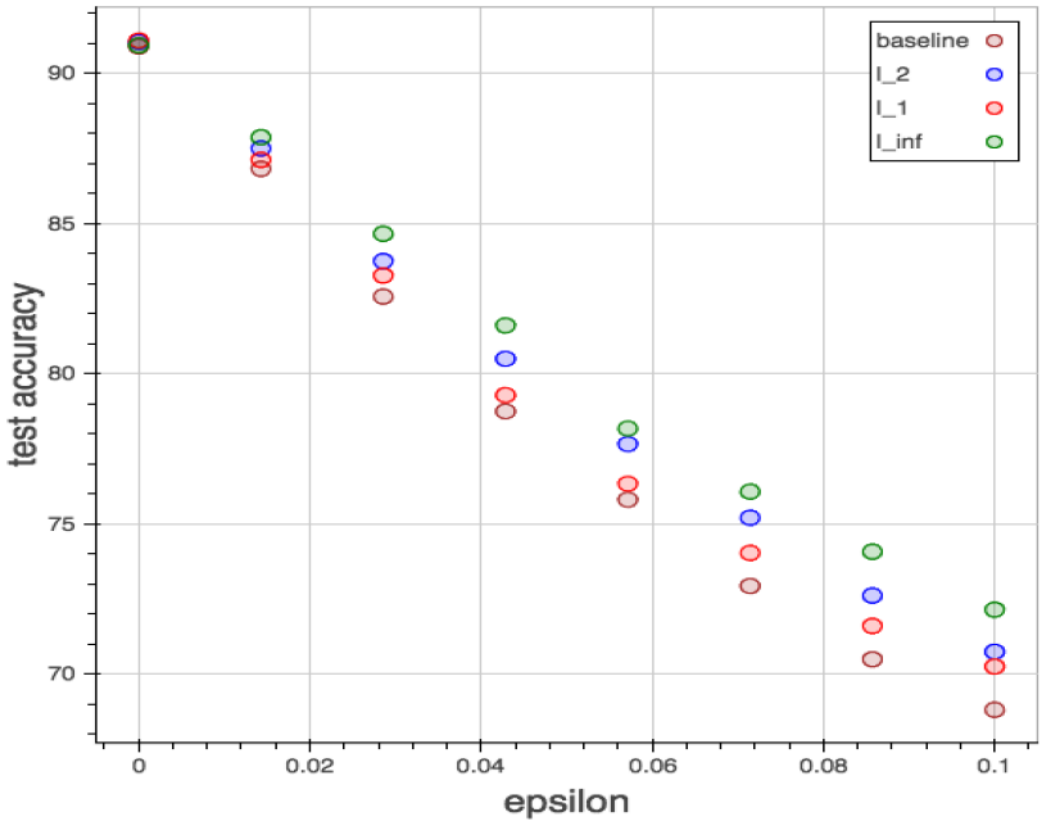
Strengths of adversarial training

(Shaham et al. 2015)

- It is a **simple** method to use and implement
- It fits into **general** frameworks
- It can be adapted to many attacks
- It can work as data augmentation and improve standard accuracy
- It has strong ties to *Robust Optimization* which could lead to some extent to theoretical guaranties

Performance of adversarial training

(Shaham et al. 2015)



Performance on CIFAR-10 under different adversarial training constraints. Test examples are generated under l_∞ perturbations.

Issues with adversarial training

(Sharma & Chen, 2017)

- It is essentially a **reactive method** : the network learns to resist a specific attack setting, but the attacker can change the attack setting.

Ex: Networks trained against l_∞ examples are vulnerable to examples created with different distortion metrics.

Issues with adversarial training

(Tramèr et al. 2017)

- Adversarial learning does not scale well to large datasets like ImageNet.

To alleviate that problem the authors propose **Ensemble adversarial training** : generate more adversarial training examples faster by *attacking several different models and transferring the examples*.

Bypassing adversarial training

(Tramèr et al. 2017)

- Adversarial training can be bypassed with **two-step attacks**, or with a one step-attack on a **randomly disturbed input**.

Table 2: **Error rates (in %) for Step-LL, R+Step-LL and a two-step Iter-LL on ImageNet.** We use $\epsilon = 16/256$, $\alpha = \epsilon/2$ on 10,000 random test inputs. R+FGSM results on MNIST are in Table 7.

	v4	v3	$v3_{adv}$	IRv2	$IRv2_{adv}$	v4	v3	$v3_{adv}$	IRv2	$IRv2_{adv}$
Step-LL	60.2	69.6	26.6	50.7	21.4	31.0	42.7	9.0	24.0	5.8
R+Step-LL	70.5	80.0	**64.8**	56.3	37.5	42.8	57.1	**37.1**	29.3	15.0
Iter-LL(2)	**78.5**	**86.3**	~~58.3~~	**69.9**	**41.6**	**56.2**	**70.2**	~~29.6~~	**45.4**	**16.5**
	Top 1					**Top 5**				

Parseval networks

(Cisse et al. 2017)

Alternatively, can we enforce robustness with more mathematically-grounded regularization ?

If every layer is a 1-Lipzschitz operation, then the entire network is → small perturbations can't have much effect.

Cisse et al. (2017) suggest **Parseval networks**, a class of networks where all weight matrices are "orthonormal", i.e. preserve input norm.

Parseval networks

(Cisse et al. 2017)

The metric uses **Signal-Noise Ratio** : $SNR = 20 \log_{10} \frac{\|x\|_2}{\|\epsilon\|_2}$

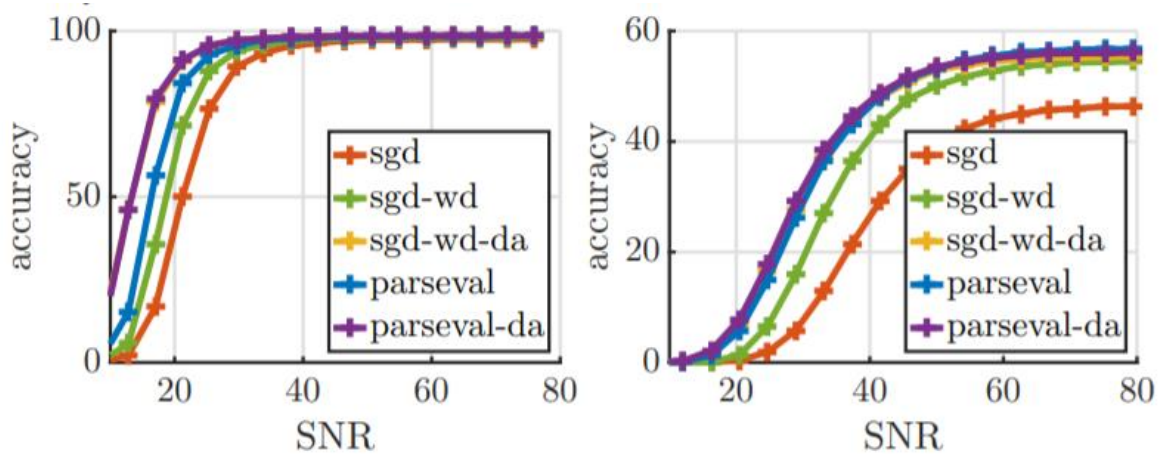


Figure 3. Performance of the models for various magnitudes of adversarial noise on MNIST (left) and CIFAR-10 (right).

Combining Parseval and adversarial training improves performance.

Parseval networks

(Cisse et al. 2017)

But optimization on that matrix manifold requires some approximations → no guaranties against strong attacks (Yan et al. 2018)

Plus, the authors do not try it on large datasets like ImageNet.

Obfuscation

Distillation

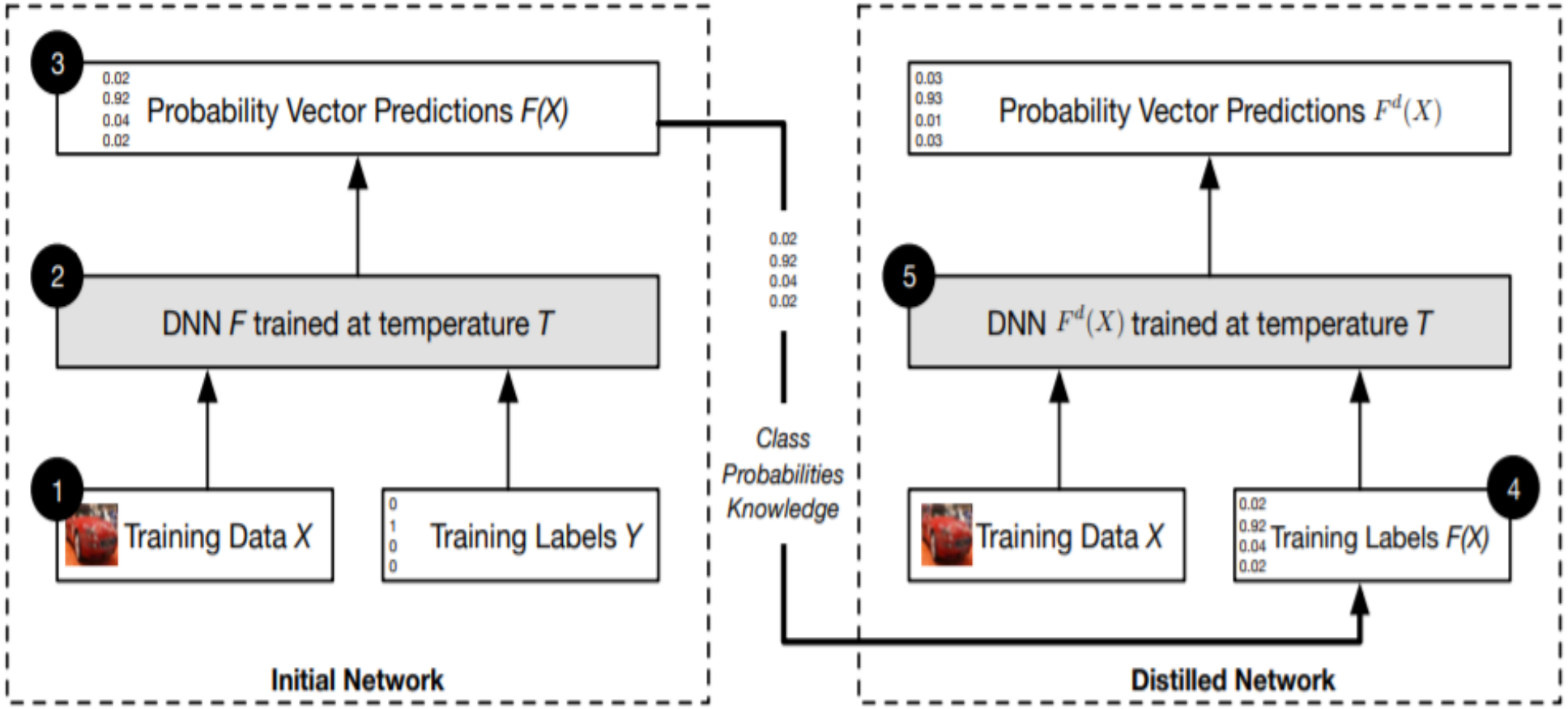
(Hinton et al. 2015)

Definition : Use the output of one or several trained model(s) as a supervision (with continuous logits) for a new model

Distillation can be used to compress a model without damaging its performance, or to ensemble models and improve overall performance (Kim et al. 2016)

Distillation procedure

(Hinton et al. 2015)



Distillation procedure

(Hinton et al. 2015)

Results get better by smoothing the softmax with a *temperature* parameter T :

$$F(X)_i = \frac{e^{\frac{Z(X)_i}{T}}}{\sum_j e^{\frac{Z(X)_j}{T}}}$$

both for the supervision and the distilled model training.

Defensive distillation

(Papernot et al. 2016)

Distillation can be used as a defense method against adversaries !

Idea : models trained on soft inputs should generalize better to points outside of the training manifold (i.e. adversarial examples ?)

Distillation vs defensive distillation

(Papernot et al. 2016)

Main difference in procedure : the model architecture stays the same for the distilled network (no compression objective).

Using a temperature remains useful.

The authors also prove that distilled model have improved *model stability* : changing a few points of the training set has little effect on the distilled model parameters.

Thermometer encoding

(Buckman et al. 2018)

Goodfellow et al. (2014) suggest that adversarial examples derive from the local linearity of neural networks in high dimension : for a weight vector w of size n :

$$w^T(x + \epsilon) = w^T x + w^T \epsilon$$

If the average magnitude of an element of w is m then the activation will grow by ϵmn which can be large even for small ϵ.

Thermometer encoding

(Buckman et al. 2018)

We can break that local linearity by reencoding the inputs into a binary representation :

For a scalar input x to a layer with value in $[0,1]$ its l-level thermometer encoding is $\tau_l(x) \in \{0,1\}^l$ such that $\tau_l(x)_k = 1_{x \geq \frac{k}{l}}$

Ex : with $l = 5, 0.7 \rightarrow 11100$

The network is then sublinear in x.

Thermometer encoding

(Buckman et al. 2018)

Note : thermometer encoding is not differentiable.

→ To attack it, instead of standard gradient descent we have to use discrete approximations of it (Discrete gradient descent, Projected gradient descent,…)

Models using thermometer encoding are shown to be more robust to attacks.

Non-differentiable models

(Tramèr et al. 2017)

Using non-differentiable models instead of neural networks can protect against gradient-based attacks

Examples :

- Nearest neighbors classifiers
- Random Forests
- …

Stochastic activation pruning

(Dhillon et al. 2018)

Apply some dropout at inference time on activations of the network.

Idea : computing the gradient of a randomized output is not meaningful → cannot generate good adversarial examples

Gradient obfuscation

(Athalye et al. 2018)

More generally, methods that do not really change the model structure but affect gradient availability/meaningfulness are called *gradient obfuscation methods.*

That includes :

- *Shattered gradients* : gradients are nonexistent or incorrect
- *Stochastic gradients* : gradients are meaningless because of randomness in the model
- *Vanishing/exploding gradients* : increase depth of networks through multiple iterations, up to the point when gradients are too small/large to be meaningful

Gradient obfuscation

Obfuscation is not a good thing !

For example, they are vulnerable to transferred inputs.

Gradient obfuscation

Procedure to bypass obfuscation (Black box setting)

- Model A has been secured with an **obfuscation method**
- Model B has been **normally trained** on the same data
- It will be hard to generate examples by attacking model A, *but…*
- Adversarial examples generated by **attacking model B** will be **effective against model A** !

In other words, black box attacks succeed where white box attacks fail.

Gradient obfuscation

(Athalye et al. 2018)

Thermometer encoding revisited :

The original justification is to break local linearity of neural networks.

In practice, the authors show that the phenomenon at hand is gradient shattering : the accuracy drops when facing black box attacks.

Gradient obfuscation

(Papernot et al. 2016)

Distillation revisited :

By training on smooth labels, one enforces smoother models hence smaller gradients (form of vanishing gradients). It does *not* affect the model sensitivity to adversarial examples.

Distillation is also vulnerable to black box attacks.

Signs of obfuscation

In fact, a very large amount of proposed defenses against adversarial attacks rely on obfuscation-related phenomena (including in recent years)

Athalye et al (2018) suggest criteria to recognize obfuscation-based methods.

Gradient obfuscation

(Athalye et al. 2018)

- Black-box attacks are better than white-box attacks
- One-step attacks perform better than iterative attacks
- Unbounded attacks do not reach 100% success
- Random sampling finds adversarial examples
- Increasing distortion bound does not increase success

Gradient obfuscation

(Athalye et al. 2018)

Those are signs that “something is wrong”.

The same authors also provide solid tools to bypass these methods without having to use black box attacks.

Gradient obfuscation

(Athalye et al. 2018)

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**

Gradient obfuscation

(Papernot et al. 2016)

What about adversarial training ?

It is partially vulnerable to black box attacks but not enough to suspect obfuscation plays a large role.

It is not an obfuscation-based method according to Athalye et al. (2017)

Denoising

Defensive Denoising

An adversarial perturbation is carefully crafted noise.

Denoising : Trying to **recover the original input** from the noisy one, before feeding it to the classifier

It can be seen as a **preprocessing** step.

Strenghs of defensive denoising

- Often easy to implement
- Can draw inspiration from a lot of previous work on denoisers
- Largely model-agnostic
- Obfuscation-free

Weaknesses of defensive denoising

- Very dataset-dependent, some methods work only on MNIST
- Performance limited by the reconstruction loss → small noise may be ignored
- Can be attack-dependent in practice

Defensive Filtering

(Osadchy et al. 2017)

Example : apply classical filters to images can eliminate adversarial noise.

Ex : Threshold on pixel values, median filter, averaging filter, Gaussian low-pass filter,…

But it mainly works on simple black/white datasets (MNIST), not so well on large ones (ImageNet)

Feature maps denoising

(Xie et al. 2019)

Within a convolutional model, add *denoising blocks* : denoising operation (mean, filter, …) + 1x1 convolution + residual connection.

Similar to the previously mentioned preprocessing steps, but here integrated to the network's structure (at different layers) hence more robust.

The authors show partial success against strong attacks.

Autoencoders

(Gu and Rigazio 2014)

Train a regular *autoencoder* to match adversarial examples to the original examples.

Problems :

- Needs extensive knowledge of the adversarial data distribution → *reactive* method
- The full network (autoencoder + classifier) is even more sensitive to adversarial examples than the original classifier)
- Does not scale well to large datasets

Denoising Autoencoders

(Gu and Rigazio 2014)

Train a *DAE* : no compressive structure, but gaussian noise is added to the input before encoding.

Problems :

- ~~Needs extensive knowledge of the adversarial data distribution → *reactive* method~~
- The full network (autoencoder + classifier) is even more sensitive to adversarial examples than the original classifier)
- Does not scale well to large datasets

Contractive Autoencoders

(Gu and Rigazio 2014)

They finally add to the objective a *contractive penalty* to the layers:

$p_i = \frac{dh^{(i)}}{dx^{(i)}}$ where $x^{(i)}$ (resp. $h^{(i)}$) is the input (resp. output) of layer i.

Problems :

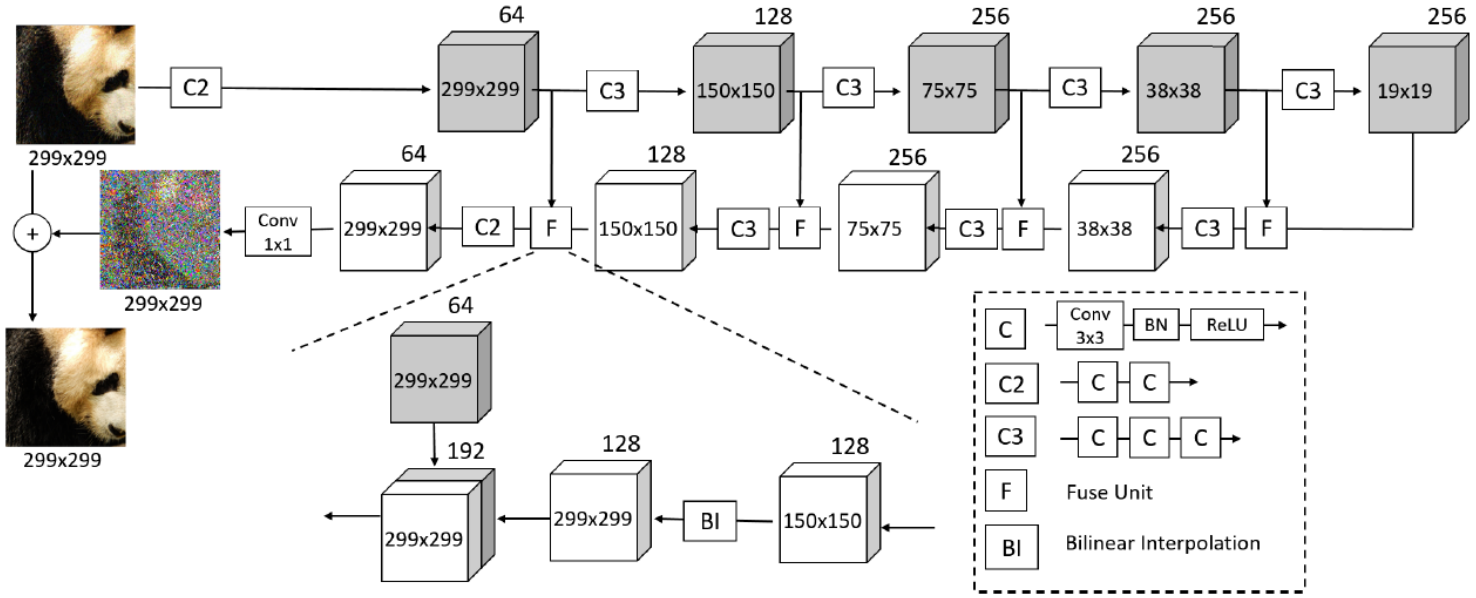
- ~~Needs extensive knowledge of the adversarial data distribution → *reactive* method~~
- ~~The full network (autoencoder + classifier) is even more sensitive to adversarial examples than the original classifier)~~ (partially solved)
- Does not scale well to large datasets

Pixel-Guided Denoiser

(Liao et. al 2017)

Extend the denoiser method to more realistic datasets than MNIST.

Solution : make the autoencoder structure more complex.



It performs better on ImageNet, *but* sensitive to small-distortion noise that affects correct classification but not reconstruction.

High level Representation-Guided Denoiser

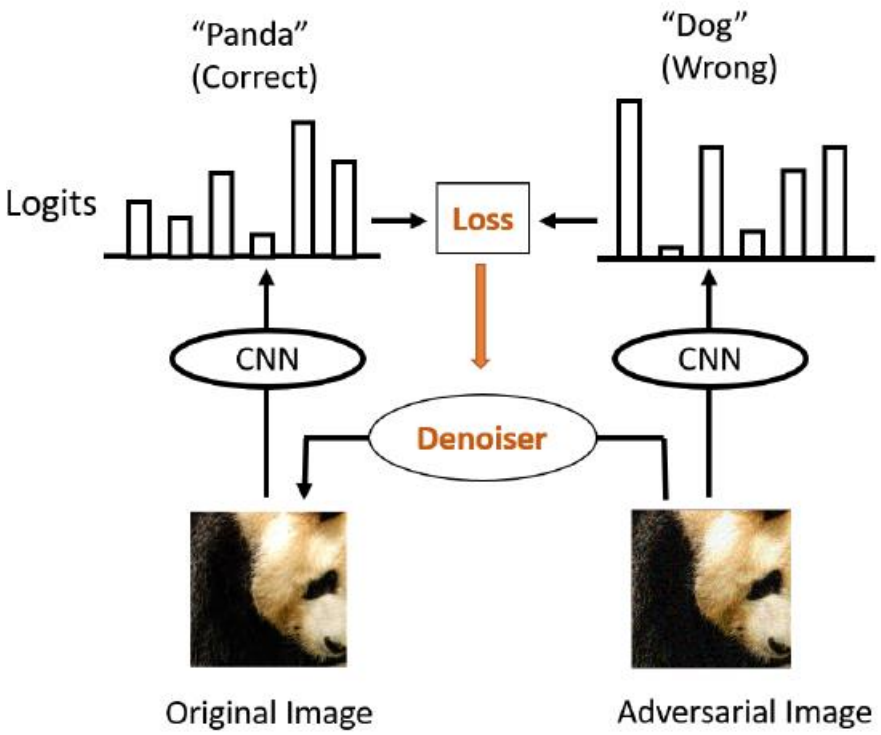
(Liao et. al 2017)

HGD strengthens the reconstruction loss of the denoiser with the **intermediate representation layers of the target model**

Specifically, for layer h_i of the classifier add a penalty term :

$$\left\| h^{(i)}(x) - h^{(i)}\left(denoiser(x_{adv})\right) \right\|_1$$

to the reconstruction loss



High level Representation-Guided Denoiser

(Liao et. al 2017)

HGD loses implementation simplicity and model agnosticity (it is designed for feed-forward networks), but we gain :

- A fairly general defense method
- Working on difficult datasets
- Robust to black-box attacks

High level Representation-Guided Denoiser

(Liao et. al 2017)

Table 3: The classification accuracy on test sets obtained by different defenses. NA means no defense.

Defense	Clean	WhiteTestSet		BlackTestSet	
		$\epsilon = 4$	$\epsilon = 16$	$\epsilon = 4$	$\epsilon = 16$
NA	76.7%	14.5%	14.4%	61.2%	41.0%
PGD	75.3%	20.0%	13.8%	67.5%	55.7%
ensV3 [31]	**76.9%**	69.8%	58.0%	72.4%	62.0%
FGD	76.1%	73.7%	67.4%	74.3%	71.8%
LGD	76.2%	75.2%	69.2%	**75.1%**	**72.2%**
CGD	74.9%	**75.8%**	**73.2%**	74.5%	71.1%

High level Representation-Guided Denoiser

(Liao et. al 2017)

However, it was only tested against **simple attacks** (FGSM variants)

HGD has since been proven vulnerable to stronger attacks like C&W (Uesato et al. 2018) or PGD (Athalye and Carlini 2018) and reaches little to zero accuracy against those.

DefenseGAN

(Samangouei et al. 2018)

Idea : defend with a generative model

GANs are trained to emulate the true data distribution.

Therefore they can be useful to project a noisy (adversarial) instance onto the true distribution.

DefenseGAN

(Samangouei et al. 2018)

The DefenseGAN procedure

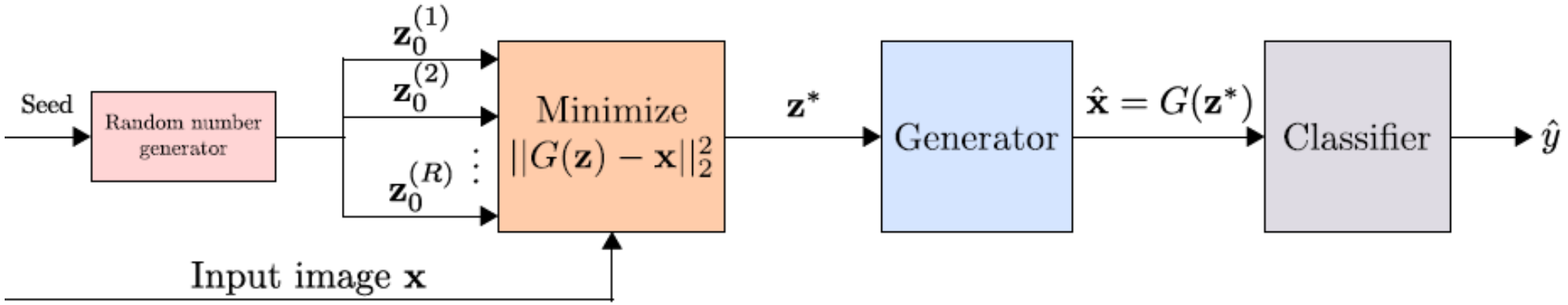
- Train a classifier C
- Train a GAN G - specifically, a Wasserstein GAN (Arjovsky et al. 2017) to generate data points from random samples z
- At test time, given an instance x :
  - Randomly generate z
  - Find with gradient descent z^* that minimizes $\|G(z^*) - x\|_1$
  - Return C $(G(z^*))$

$G(z^*)$ is the projection of (potentially adversarial) x onto the true data manifold

DefenseGAN

(Samangouei et al. 2018)

The DefenseGAN pipeline



Breaking DefenseGAN

DefenseGAN is heavy in implementation, but effective for attacks crafted against C.

But what about targeting the full network $C(G(.))$?

Breaking DefenseGAN

DefenseGAN is heavy in implementation, but effective for attacks crafted against C.

But what about targeting the full network $C(G(.))$?

This is difficult because the full network is very deep and has vanishing gradients…

…but that is just obfuscation !

→ It can be circumvented

Breaking DefenseGAN

(Athalye et al. 2018)

DefenseGAN can be attacked in a white-box setting.

Idea : use gradient attacks on $y = C(G(z))$ but use $\frac{dy}{dG(z)}$ as a substitute for $\frac{dy}{dz}$

This is an instance of a method called Backpropagation Differentiable approximation (BPDA)

Breaking DefenseGAN

(Athalye et al. 2018)

Defense	Dataset	Distance	Accuracy
Buckman et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%*
Ma et al. (2018)	CIFAR	0.031 (ℓ_∞)	5%
Guo et al. (2018)	ImageNet	0.005 (ℓ_2)	0%*
Dhillon et al. (2018)	CIFAR	0.031 (ℓ_∞)	0%
Xie et al. (2018)	ImageNet	0.031 (ℓ_∞)	0%*
Song et al. (2018)	CIFAR	0.031 (ℓ_∞)	9%*
Samangouei et al. (2018)	MNIST	0.005 (ℓ_2)	55%**

It however seems to be only partly successful.

PuVAE

(Hwang et al. 2019)

Introduced recently as an alternative to DefenseGAN.

The idea and induced robustness are similar, but :

- The manifold is represented with a Variational AutoEncoder
- Each label class has its own encoded manifold, and the defense checks which is closer to the input
- PuVAE is *much* (over 100 times) faster to run at inference time.

Renoising

Renoising

Adversarial perturbation : carefully crafted noise

Renoising : adding noise to an input/intermediate output of the algorithm to “distort” the adversarial noise

Noise can be useful both at training and inference time

Renoising

Examples :

- Lyu et al. (2015) : add gaussian noise after each convolution layer in an AlexNet network at training and inference
- Xie et al. (2018) : resize inputs to random sizes, then randomly pad in every direction with zeros.

Renoising vs obfuscation

Is renoising obfuscation ?

It can be - Random rescaling is according to Athalye et al. (2018)

One should be careful to check obfuscation phenomena by trying renoising methods against black box attacks

But there are other reasons why adding noise is helpful in increasing robustness.

Renoising vs obfuscation

In fact, it can be proven that to some extent adding noise at inference time likely erases **any adversarial example** (Lecuyer et al. 2018, Pinot et al. 2019) with a certain probability.

Various work (Fawzi et al, 2016 ; Lecuyer et al, 2018 ; …) have provided robustness bounds, drawing inspiration from frameworks such as Differential privacy.

We will explore one of them.

Defense Guaranties

(Pinot et al. 2019)

General setting : applying to any layer of a neural network a noise from the Exponential family (Gaussian, Laplace, Uniform, …)

Neural networks $\mathcal{X} \rightarrow \mathcal{Y}$ with noise are *probabilistic mappings* :

$$M : x \in \mathcal{X} \rightarrow p$$

(p a probability measure over $\mathcal{Y}$)

We also have a probability distribution over the set of inputs : $(x, y) \sim \mathcal{D}$ with marginal $\mathcal{D}_{\mathcal{X}}$ over $\mathcal{X}$

Defense Guaranties

(Pinot et al. 2019)

Adversarial robustness can be quantified with *probability-change risk* within a radius :

$$PCRisk_{\alpha}(M, \epsilon) = \mathbb{P}_{x \sim \mathcal{D}_X}\left[\exists \tau \in \mathcal{B}(\alpha), d\left(M(x+\tau), M(x)\right) > \epsilon\right]$$

i.e. for some perturbation $|\tau| < \alpha$ of the input, the output distributions diverge by ϵ or more, according to some divergence metric d.

Ex : $d =$ KL-Divergence, TV-Distance, etc.

The authors use the Renyi distance which generalizes KL-Divergence.

Defense Guaranties

(Pinot et al. 2019)

They prove that a network with random noise is *robust* (i.e. has small PCRisk) for a value of ϵ depending on radius α, the sensitivity of the layers *before the noise injection*, and properties of the noise family.

Defense Guaranties

(Pinot et al. 2019)

Ex : for Gaussian noise $(0, \Sigma)$,

$$\epsilon = \frac{\lambda \Delta_{\alpha}(\phi)^2}{2\sigma_{min}(\Sigma)}$$

With λ parameter of the Renyi divergence (1 for KL-Divergence), ϕ the pre-noise subnetwork, and

$$\Delta_{\alpha}(f) = \sup_{x,y \in \mathcal{X}, \|x-y\| < \alpha} \|f(x) - f(y)\|_2$$

Defense Guaranties

(Pinot et al. 2019)

Q : how will the noise affect accuracy ?

The authors prove a relation at fixed noise between the accuracy under attack and the accuracy without attack :

$$|Risk(M) - Risk_{\alpha}(M)| < 1 - e^{-\epsilon}\mathbb{E}_{x}\left[e^{-H\left(M(x)\right)}\right]$$

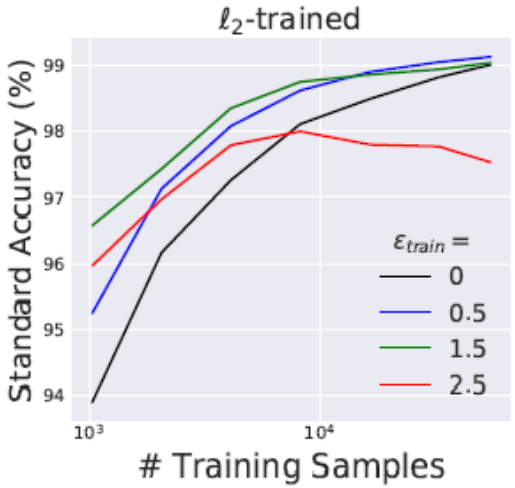
with H the Shannon entropy.

But this does not state how the risk changes with the noise…

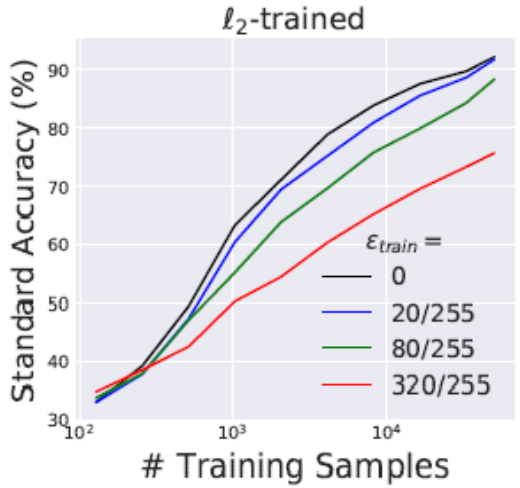
Robustness vs accuracy

(Tsipras et al. 2018)

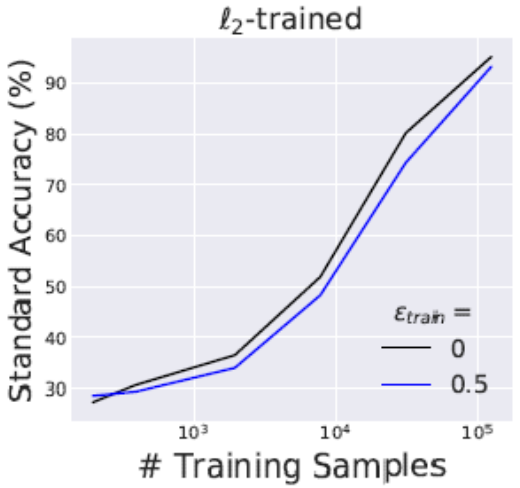
It may be a general rule that robustness and accuracy are incompatible goals.



(a) MNIST



(b) CIFAR-10



(c) Restricted ImageNet

Those phenomena are not fully understood at this point.

Detection

Detection

We have seen methods to :

- Change/train the model to classify correctly adversarial instances
- Preprocess the instances beforehand without altering the model

But both challenges have proven very difficult.

Alternative idea : learn how to detect instances as adversarial

Advantages of Detection

- Can be conceived separately from the model
- Eases the burden on the model representation
- Intuitively “should be possible” : since adversarial examples are misclassified they must exhibit some differences from regular instances (Grosse et al. 2017)

Statistical tests

(Grosse et al. 2017)

Apply two-sample statistical hypothesis testing to the detection of adversarial instances.

Statistical tests

(Grosse et al. 2017)

Let $X_1, X_2 \in \mathcal{X}$ two instances, drawn from two distributions
$\mathrm{X}_1 \sim p, X_2 \sim q$
We have two hypothesis $H_0: p = q$ and $H_A: p \neq q$

We can define a statistical test $\mathcal{T}: \mathcal{X}^{\mathrm{m}}{}_{\mathrm{X}}\mathcal{X}^{n} \rightarrow \{0,1\}$ under H_0, and compare its p-value to a previously fixed threshold α (typically 0.01 or 0.05) to reject or not the hypothesis.

Statistical tests

(Grosse et al. 2017)

What test $\mathcal{T}$ should we apply ?

For high dimensionality the authors recommend an estimator of Maximum Mean Discrepancy :

$$MMD_b(\mathcal{F}, X_1, X_2) = \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} f\left(x_{i,1}\right) - \frac{1}{n} \sum_{i=1}^{n} f\left(x_{i,2}\right)$$

with $\mathcal{F}$ a class of kernel functions.

Statistical tests

Q : When can we actually apply that method ?

I.e. what is the threat model ?

Statistical tests

A : When we have :

- a set A of natural instances
- a set B of instances, either all natural or all adversarial (from the same adversarial distribution)
- We want to know if the instances in B are natural or adversarial

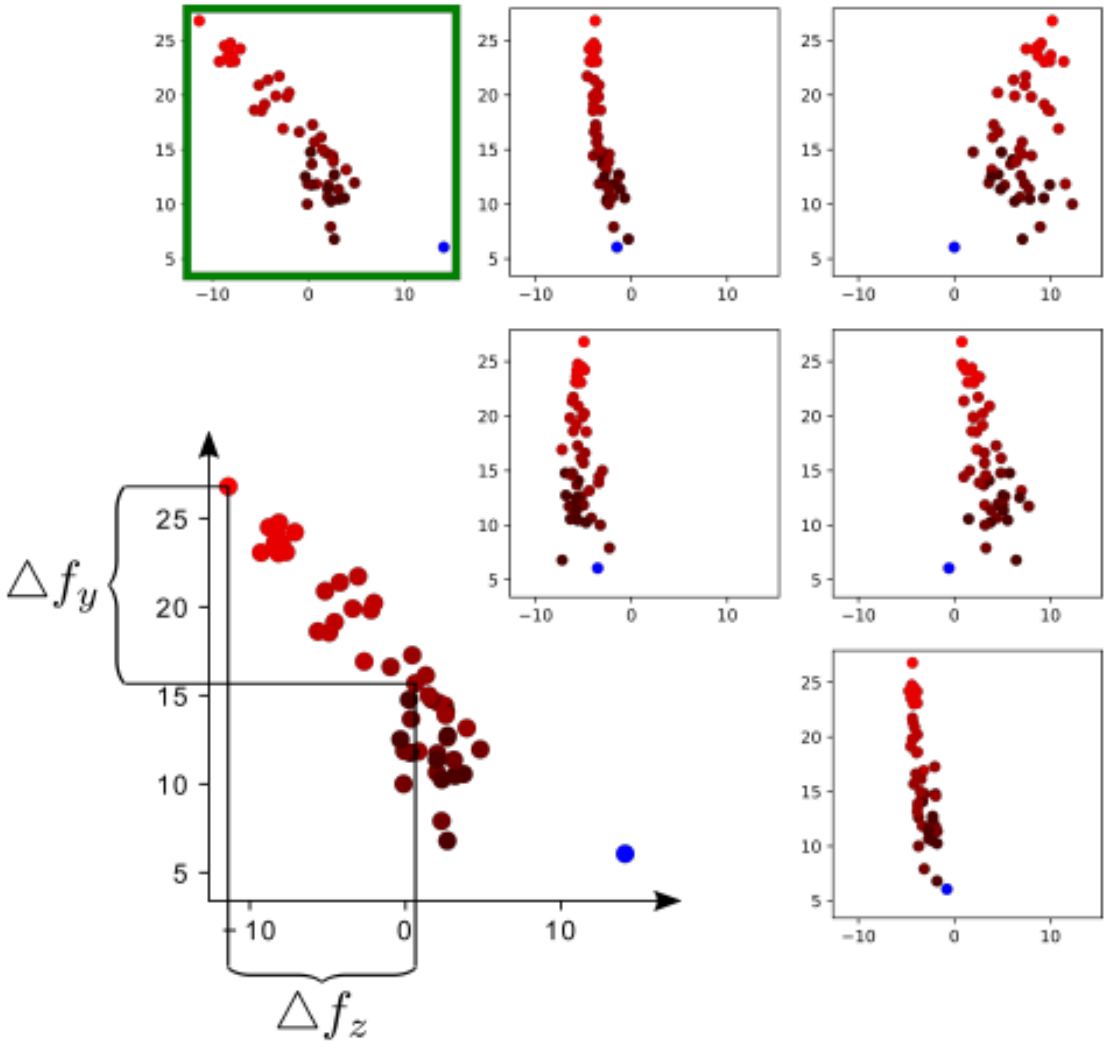
→ The test is interesting in theory but not very useful for practical applications (instance-wise).

Statistical tests

(Roth et al. 2019)

Simpler tests on the network layers (especially the final layer of log-probabilities for each class).

They claim that logits output for adversarial instances have by nature very **clear directions of variation.**



Statistical tests

(Roth et al. 2019)

The quantity :

$$g_{y^*}(x,\eta) = f_{y*}(x+\eta) - f_{y*}(x) - \max_{y \neq y^*} f_y(x+\eta) - f_y(x)$$

is greater for adversarial instances x, where $f_y(x)$ is the log-probability predicted for instance x and class y and y^* is the predicted class for x.

They compute the value of $g_{y^*}(x,\eta)$ normalized over noise η and compare it to a threshold τ to determine if x is adversarial.

Statistical tests

Other collected statistics looked at in literature to distinguish adversarial and natural examples have included :

- PCA coefficients of the input (Bhagoji et al. 2017)
- PCA of intermediate convolutional filters (Li et al. 2016)
- Minimal/maximal/percentile values of intermediate convolutional filters (Li et al. 2016)
- Kernel density estimation on the final representation layer (Feinman et al. 2017), to evaluate the distance of an instance to the training data manifold.

Classifying outliers

Alternatively, assuming adversarial and natural examples differ, one can simply train a separate classifier to distinguish between the two !

General static procedure :

1. Train a classifier C on natural instances
2. Generate adversarial examples from C
3. Train a detector D on the natural and adversarial examples

Classifying outliers

(Metzen et al. 2017)

General dynamic procedure :

1. Train a classifier C on natural instances
2. Generate adversarial examples from C
3. Train a detector D on the natural and adversarial examples
4. Generate adversarial examples from C and D
5. Repeat 3 and 4 until some convergence criterion is reached

Dynamic training is meant to mitigate the effects of the attacker knowledge of the detector

Binary classification of inputs

(Gong et at. 2017)

Train a classifier C, then a simple binary detector D on the natural and adversarial test sets.

The authors train in a static setting, *but* do apply a second round attack, i.e. :

From an adversarial instance x_{adv} generated against C, generate a new x_{adv2} aiming to fool D. That new instance does not fool C anymore, i.e. we cannot fool both network simultaneously.

Classification of intermediate outputs

(Metzen et al. 2017)

Use different small subnetworks branching from different intermediate layers of the network.

Subnetworks on intermediate layers seem to provide better detectability than on early or final layers.

They train in static and dynamic settings.

Extra outlier class

(Grosse et al. 2017)

Use the same network as classifier and detector

→Add an extra “outlier” label to the original classes, and train on a natural + adversarial dataset.

This method has a small negative effect on performance.

Classifying outliers

Pros :

- No need to manually engineer statistics
- Leverage the strong representation power of machine learning algorithms

- Cons :
- The detector itself may be vulnerable to attacks
- In-nature **reactive** : implies the attack scheme is known in advance

MagNet

(Meng et al. 2017)

MagNet uses a detector that does not learn on adversarial examples.

Instead, a neural network will attempt to learn the real data manifold, to then use distance to the manifold as a metric

(i.e. uses a network to do the work of a statistical metric).

This is done with an autoencoder that represents the original data manifold.

MagNet

(Meng et al. 2017)

Two possible metrics for the detector :

- Use the reconstruction loss

$$\left\| dec\big(enc(x)\big) - x \right\|_2$$

- Use the final hidden layer Z of the target classifier

$$\left\| Z\left(dec\big(enc(x)\big)\right) - Z(x) \right\|_2$$

The autoencoder can also be used as a reformer to correctly classify adversarial examples.

Limits of detection

(Carlini and Wagner 2017)

The previous detection methods are weaker than expected!

Several are dataset-dependent (MNIST) : PCA,…

Many cannot resist to a strong attack like C&W (Grosse et al, Gong et al) in a dynamic setting (Metzen et al).

About *all* can be broken by designing a specific loss function targeting each defense.

Limits of detection

(Carlini and Wagner 2017)

Example : Feinman et al. (2017) use a kernel density estimator on the final hidden layer H_{n-1} to discriminate adversarial examples :

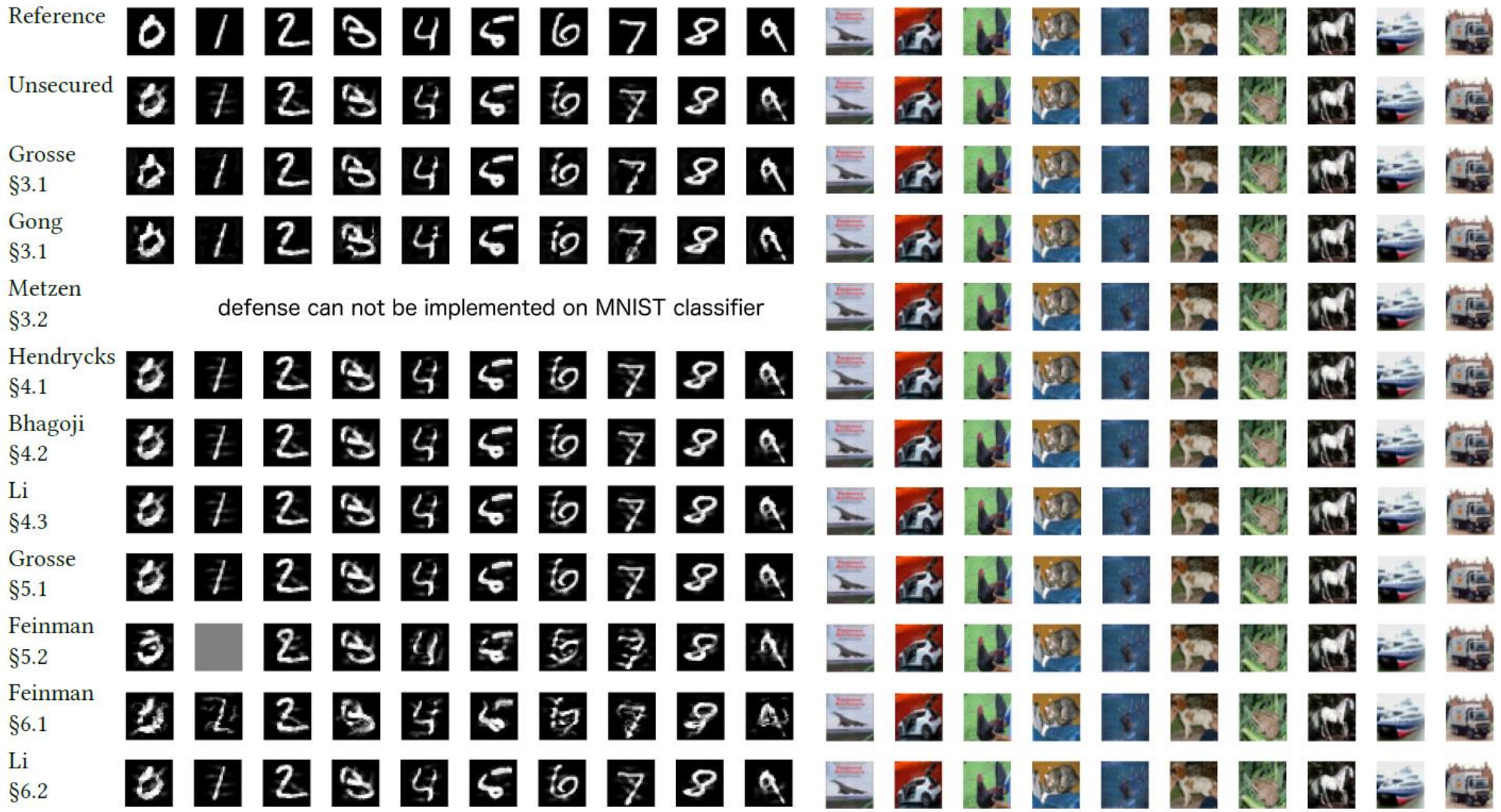
$$KDE(x^*) = \frac{1}{\mathcal{X}_T} \sum_{x \in \mathcal{X}_T} \exp\left(\frac{|H_{n-1}(x^*) - H_{n-1}(x)|^2}{\sigma^2}\right)$$

This method can be broken by computing adversarial examples with training objectives targeting that function :

$$x_{adv} = argmin_{x'} \|x' - x\|_2^2 + c.\Big(l(x') + \max\big(-\log\big(\mathrm{KDE}(\mathrm{x}')\big) + \epsilon, 0\big)\Big)$$

Limits of detection

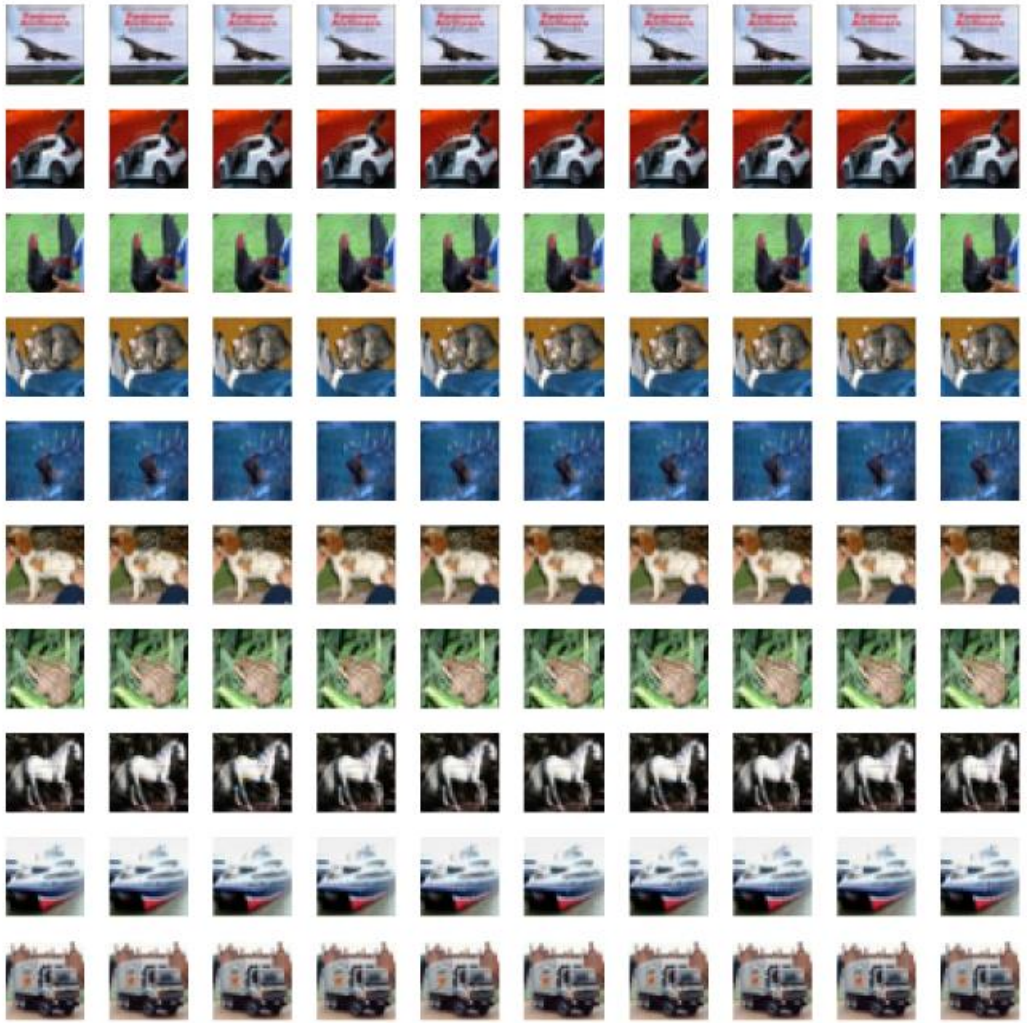
(Carlini and Wagner 2017)



Limits of detection

(Carlini and Wagner 2017)

And against MagNet :



Reverse Cross-entropy

(Peng et al. 2018)

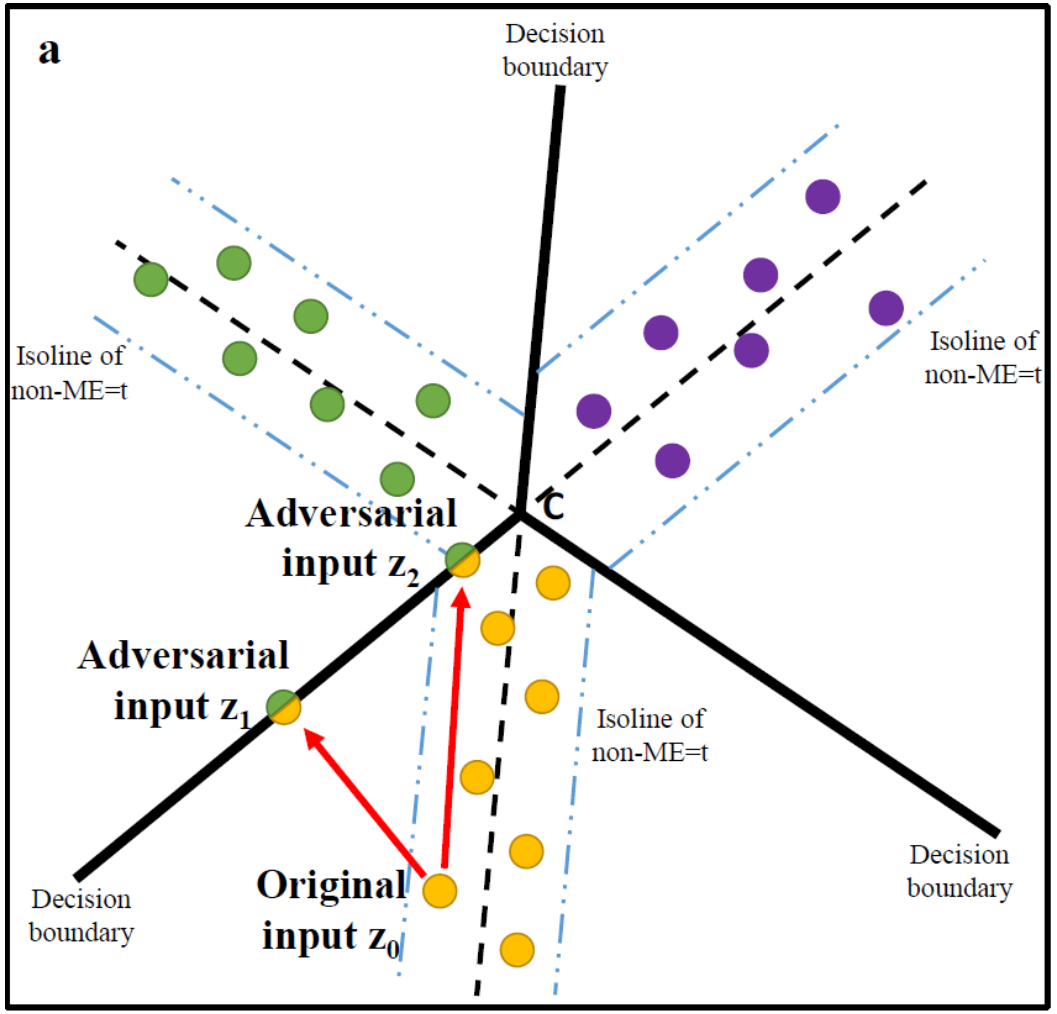
An original detection method based on an alternative training loss :
reverse cross-entropy

Reverse Cross-entropy

(Peng et al. 2018)

The authors enforce their classifier to project training instances onto a low-dimension manifold characterized as the boundary between non-maximal classes.

This enforces greater distortion to fool both the detection criterion and the classifier.



Reverse Cross-entropy

(Peng et al. 2018)

They therefore require their model to associate close logits to all non-maximal classes.

To enforce that, instead of cross-entropy loss they train their n-labels softmax classifier C to maximize the dot product :

$$L_{RCE}(x, y) = C(x)^T R_y$$

where R_y is the n-dimensional vector of value 0 at index y and 1 everywhere else.

Reverse Cross-entropy

(Peng et al. 2018)

Note that this does not maximize the logit of the correct class y, but instead *minimize* it.

Therefore, for classification at inference time they reverse the logits.

Domain-specific defenses

Use domain knowledge

Defenses we have seen so far are not fully satisfying.

Besides, we have mentioned that adversarial examples are unavoidable in a standard machine learning setting.

Use domain knowledge

Defenses we have seen so far are not fully satisfying.

Besides, we have mentioned that adversarial examples are unavoidable in a standard machine learning setting.

In that case, a possible solution is to leave that setting…

… i.e. make use of methods from other areas of knowledge.

Domain knowledge

Historically, most literature examples on adversarial attacks belong to the field of computer vision.

→Use methods based on classical image processing.

Domain knowledge : vision

(Guo et al. 2017)

Ex 1 : apply image-specific data preprocessing (aka denoising) :

- Image cropping/rescaling
- Bit-depth reduction
- JPEG compression
- Image quilting (replacing patches with others from clean images)
- …

Domain knowledge : vision

(Xu et al. 2018)

Ex 2 : apply different transformations designed to “squeeze features”, with the hope that adversarial examples will react differently to them (aka detection)

- Color depth quantization (8-bit)
- Spatial smoothing (median blur…)

Audio defenses

(Yang et al. 2018)

Audio-based methods are found less frequently in literature, but exist.

Yang et al. (2019) : simple input transformations similar to images, such as

- Quantizing the amplitude
- local smoothing

are ineffective

Unfortunately, more audio-based transformations like downsampling the audio band do not perform much better.

Cause : all of those idea rely on obfuscation.

Audio defenses

(Yang et al. 2019)

Simple input transformations similar to image defenses, such as

- Quantizing the amplitude
- local smoothing

are ineffective

Unfortunately, more audio-based transformations like downsampling the audio band do not perform much better.

Reason : all of those ideas rely on obfuscation.

Temporal dependencies

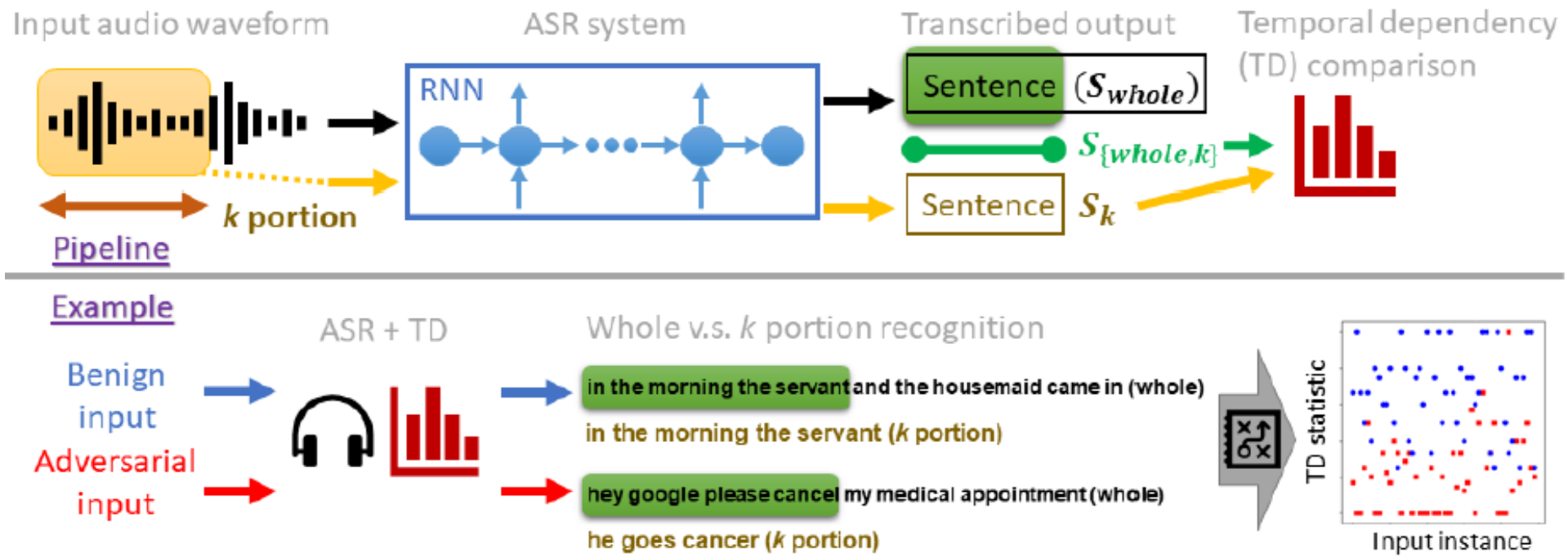
(Yang et al. 2019)

A family of detection methods, based on leveraging the **temporal dependency** of audio files.

Hypothesis : adversarial examples lose some temporal information, which can be used for detection

Temporal dependencies

(Yang et al. 2019)



The defense seems effective against strong attacks (C&W, Alzantot et al,…) and some defense-tailored attacks they suggest.

ADAGIO

(Das et al. 2019)

Adversarial Defense for Audio in a Gadget with Interactive Operations

Incorporates compression schemes as a denoising mechanism :

- MP3 compression
- AMR encoding

They apply the defense against C&W successfully, but not against tailored attacks.

Audio compression

(Rajaratnam et al. 2018a)

Apply various preprocessing operations, and as a detection criterion check whether they change the results :

- MP3 Compression
- AAC Compression
- Speex Compression
- Opus Compression
- Band-pass Filtering
- Audio Panning and Lengthening.

Audio compression

(Rajaratnam et al. 2018a)

They show that the methods work better when ensembled with Learned Threshold Voting.

The defense is successful against Alzantot et al (no adaptative attack).

Noise flooding

(Rajaratnam et al. 2018b)

Check how much noise is needed to change the prediction.

- Select a noise limit ϵ
- Add a random noise array sampled within $[-\epsilon, \epsilon]$ to the audio sample
- Compare the model result for the original and perturbed array
- If the prediction does not change repeat with a larger ϵ

Noise flooding

(Rajaratnam et al. 2018b)

The noise can be sampled in many ways (within different frequency bands) → several detectors that can be ensembled

The authors suggest to use Decision-tree classification (e.g. AdaBoost) to ensemble, because different frequency bands should be treated differently.

Conclusion

Defense desiderata (revisited)

- Robustness
  - Performance under attack compares to performance without attacks
  - Effective under different attack schemes
  - **Proactive** : effective without prior knowledge of attack strategy
  - Effective against black box attacks as well
  - Effective when the adversary is aware of the defense
- Efficient:
  - Low impact on accuracy
  - Low impact on speed
  - Low impact on existing architecture

Some takeaways to bring home

(Inspired from Carlini and Wagner 2017)

Always

- Make precise and *reasonable claims*
- Try your defenses on *several datasets of variable difficulty*
- Try your defenses on *the strongest attacks*
- Look out for *obfuscation* and *transferability*