DEPARTMENT OF
NETWORKED SYSTEMS
AND SERVICES

Certified Robustness to Adversarial Examples

Gergely Ács

CrySyS Lab, BME

acs@crysys.hu

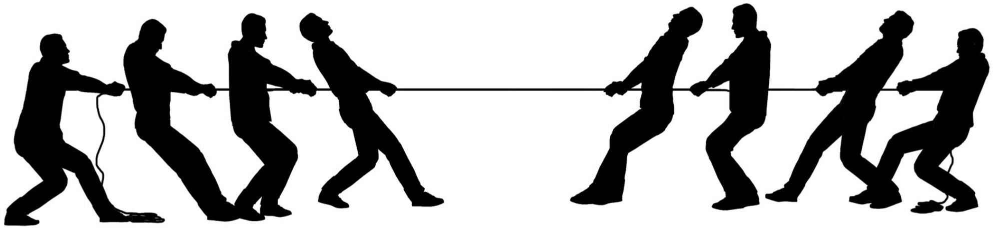
MŰEGYETEM 1782

Roadmap

- Why do we need certified robustness?
- Complete and incomplete verification
- Randomized Smoothing
- Differential Privacy and Certified Robustness
- Conclusions

WHY DO WE NEED CERTIFIED ROBUSTNESS?

“Arms race”



Attackers **Defenders**

- practitioners design new ways of hardening classifiers against existing attacks, and then a new class of attacks is developed that can penetrate this defense
- Distillation (Papernot et al., 2016) -> Broken: Carlini & Wagner, 2017
- Rotation and scaling (Lu et al., 2017) -> Borken: Athalye, 2017
- ...

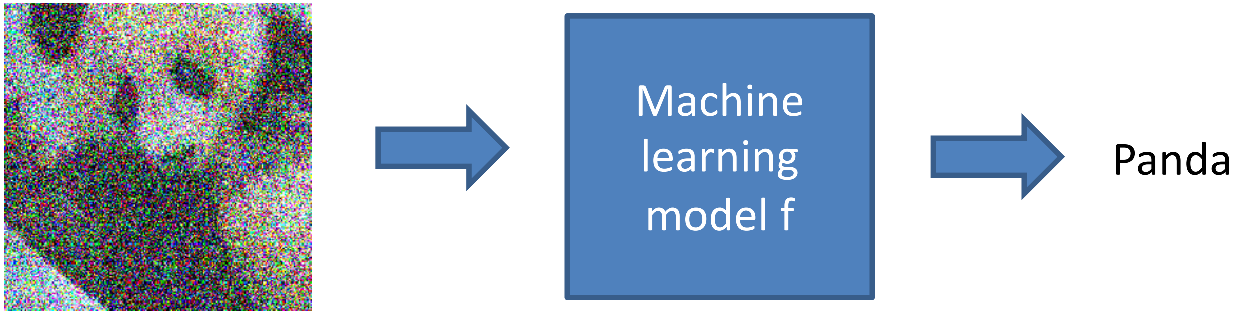
"Arms race"

- The lesson has been learnt from crypto…

- Design classifiers that are guaranteed to be robust to adversarial perturbations!!!
  - even if the attacker is given full knowledge of the classifier
  - any weaker attempt of "security through obscurity" could ultimately prove unable to provide a robust classifier

What is robustness?

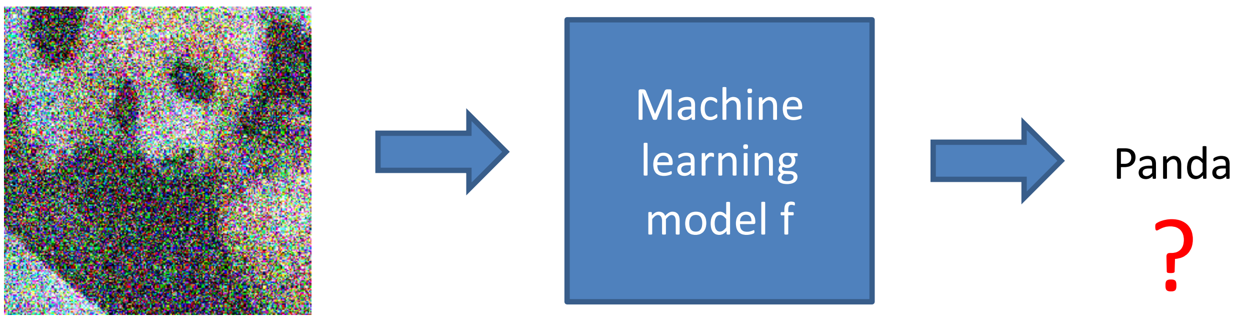
- f is *robust* to adversarial examples if its output is *insensitive* to small changes to any *plausible* input that may be encountered in deployment
  - model robustness is typically assessed on inputs from a test set that are not used in model training



- Problems:
  1. How to verify that an already trained model is robust?
  2. How to train a model so that it becomes robust?

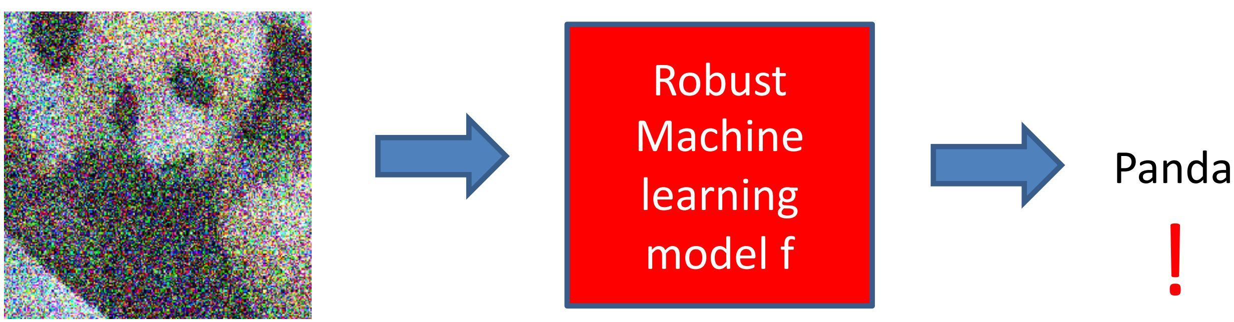
Problem of Verification

- Find the largest "neighborhood" of a sample **x** such that all neighboring samples within this neighborhood has the same prediction
  - neighboring samples are visually similar
  - a robustness verification program A gives a guarantee that no adversarial examples exist within a certain radius of **x**



Problem of Robust Training

- How to train a model so that it becomes robust?
  - a model is *robust* to adversarial examples if its output is *insensitive* to small changes to any *plausible* input that may be encountered in deployment
  - model robustness is typically assessed on inputs from a test set that are not used in model training



What do we mean by neighborhood?

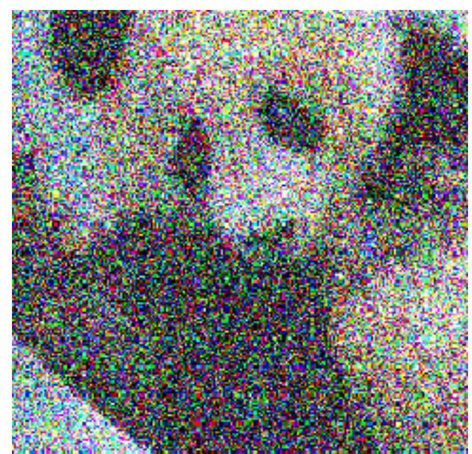
- P-norm ball or radius r:

$$B_p(r) := \{\delta \in \mathbb{R}^n : ||\delta||_p \leq r\}$$

- p = 0: changes are concentrated on a few pixels
- p > 1: change may spread out over many or all features
  - more powerful, as they can remain invisible

Original input

“L_2 neighborhood” with $\delta = 0.1$



What do we mean by neighborhood?

- *p*-norm ball or radius r:

$$B_p(r) := \{\delta \in \mathbb{R}^n : ||\delta||_p \leq r\}$$

- An attacker can craft a successful adversarial example for a given p-norm if they find $\delta \in B_p(r)$ such that $f(x) \neq f(x + \delta)$
  - an adversary can find perturbation δ so small that $x + \delta$ looks just like x to the human eye, yet the network classifies $x + \delta$ as a different, incorrect class

Original input **x**

f(**x**) = Panda

$x + \delta_1$

$f(x + \delta_1) =$ Ostrich

$x + \delta_2$

$||\delta_1||_2 \leq ||\delta_2||_2$

$f(x + \delta_2)$ = Ostrich

Robustness, once again …

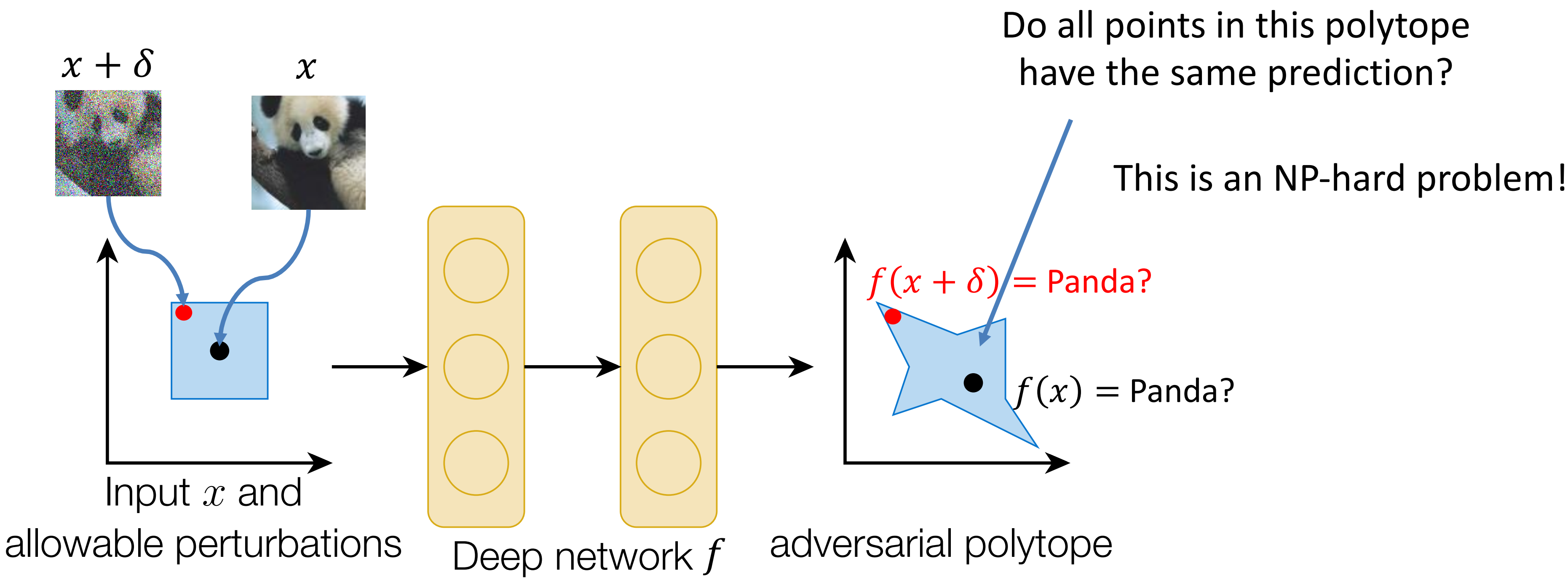
- Definition:
  f is R-robust at x, if for any $\delta \in B_p(R)$, $f(x) = f(x+\delta)$
  – f is constant around R-sized "neighborhood" of input *x*

- ***Robustness verification***
  – Is f R-robust at x?

- ***Robust training***
  – f is guaranteed to be (provably) R-robust no matter what x is

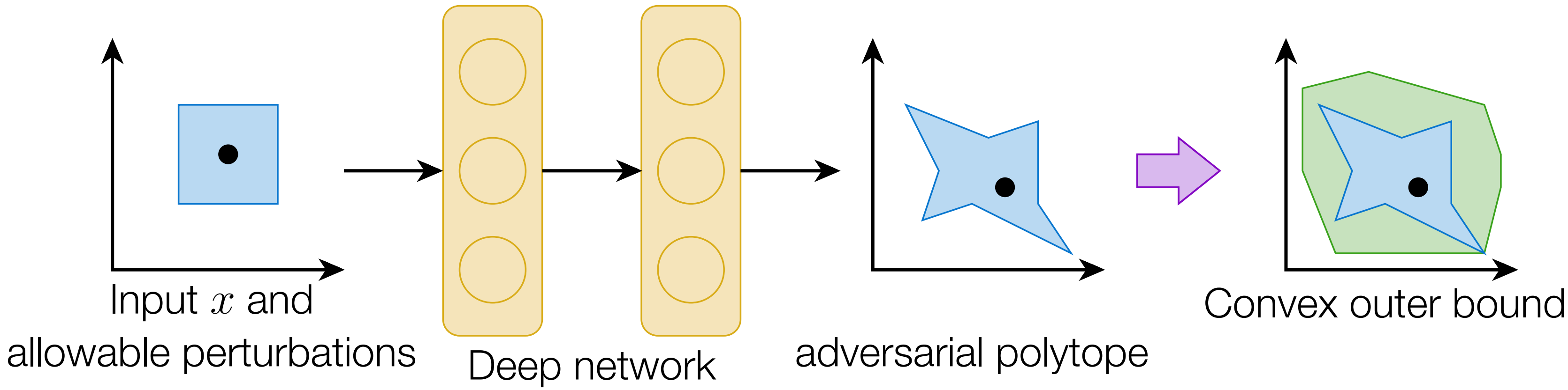
COMPLETE AND INCOMPLETE VERIFICATION

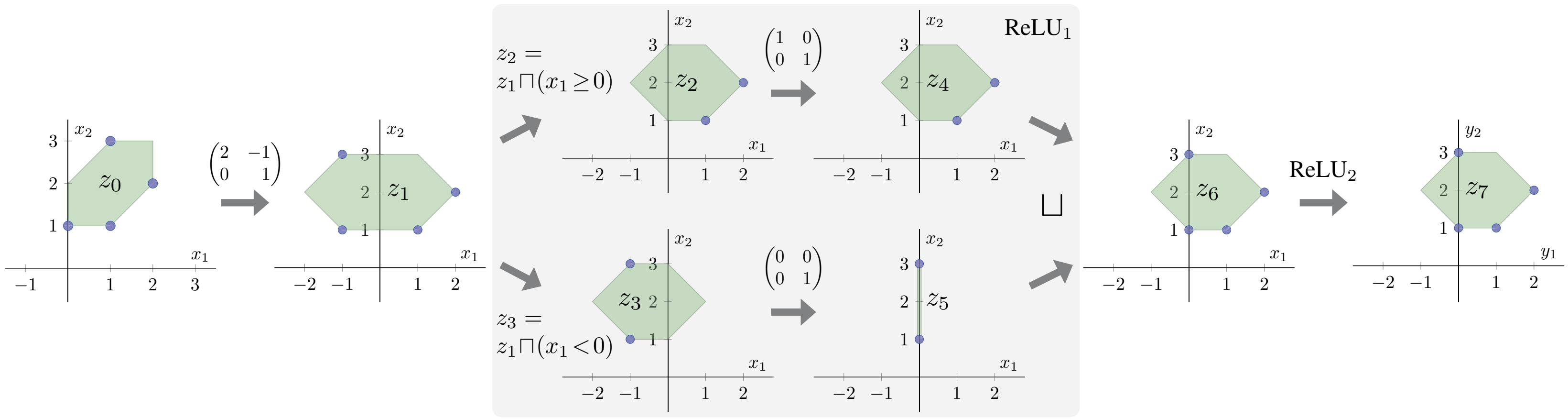
Verification: it is hard…



- **Complete verifiers** reason about the exact polytope
  - Slow, (may not terminate in reasonable time) but gives definite answer (if it terminates)
- **Incomplete verifiers** bound the adversarial polytope
  - More scalable, but provide only approximations (false negatives may occur)

Examples for incomplete (but sound) verifiers





Robust training with convex approximations

- Replace each training sample with its « worst » neighbor, then train the network with this new training data

$$\min_f \sum_{i=1}^{N} \max_{||\delta||_p \leq R} Loss(f(x_i + \delta), y_i)$$

- Computing the worst-case neighbor is hard, hence they bound *f* with a convex function

Summary

- Complete verifiers work only on very small networks
  – due to NP-hard nature of the underlying problem

- Incomplete verifiers (or training) work only on neural networks with ReLU activation functions

Is there a method for any kind of neural networks that is also scalable?

RANDOMIZED SMOOTHING

Randomized smoothing: Overview

- Train the classifier *f* with the samples corrupted by some noise with variance σ
- In the testing phase, return the class which *f* is most likely to return when *x* is corrupted by Gaussian noise with variance σ

Original input

Corruption with Gaussian noise $\sigma = 0.5$

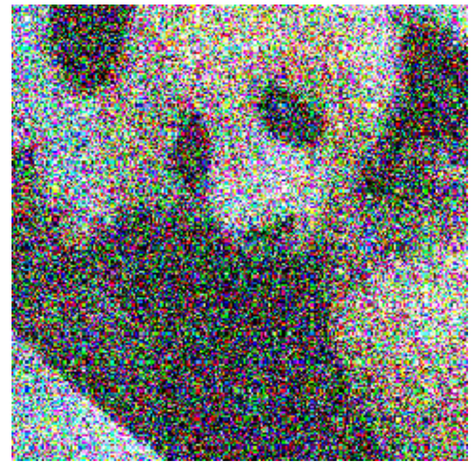
98 % panda, 2 % ostrich

Panda!!!

Randomized smoothing: Provably robust

Original input

Corruption with Gaussian noise $\sigma = 0.5$

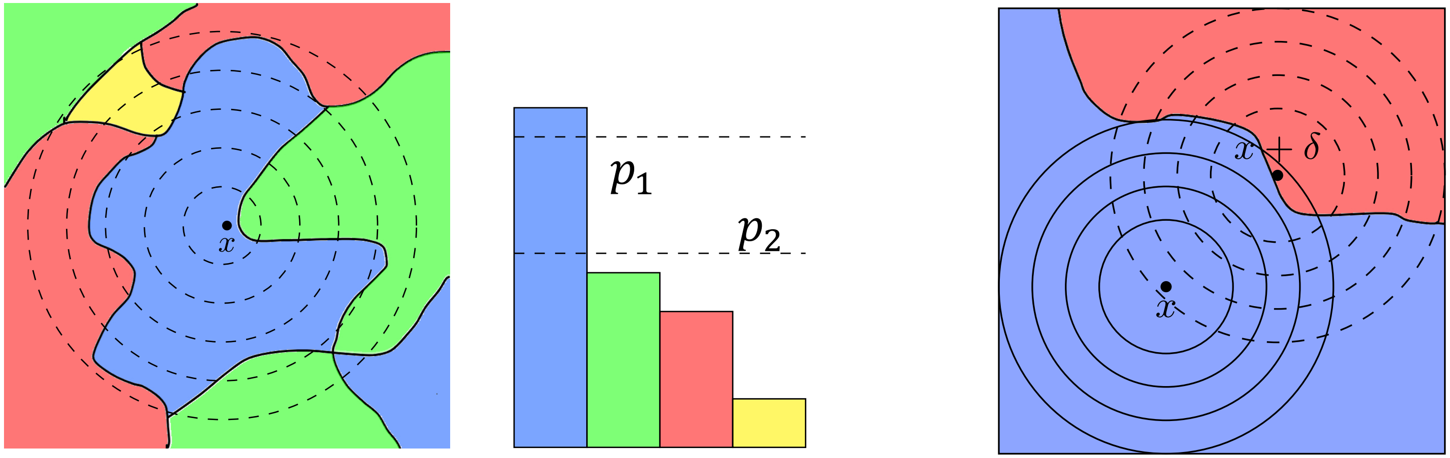


98 % panda, 2 % ostrich

Panda!!!

- This classifier is **provably robust**!
  - it has the same prediction around **any** input sample x within a L_2-radius of $\sigma \cdot \Phi^{-1}(p_1)$, where p_1 is the probability of the most confident class at sample x (0.98 for panda above)

Randomized smoothing: details



- transforms any arbitrary base classifier f into a new “smoothed classifier” g that is certifiably robust in l_2-norm

$$g(x) = \arg\max_{c \in \mathcal{Y}} \ \mathbb{P}(f(x+\varepsilon) = c)$$
$$\text{where} \ \ \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$$

- g(x) returns the most probable prediction by f of random Gaussian corruptions of x

Randomized smoothing: guarantee

- Theorem: g is R-robust at x, where $R = \frac{\sigma}{2}(\Phi^{-1}(p_1) - \Phi^{-1}(p_2))$
  - p_1 is the probability of the most likely class with f(x)
  - p_2 is the probability of the second most likely class with f(x)
  - Simple upper bound: $R \leq \frac{\sigma}{2}\Phi^{-1}(p_1)$

$\Rightarrow$ **classifier f is constant (robust) around x with radius R**

- this L_2 robustness guarantee is tight
  - it is impossible to have a guarantee larger than R (with the L_2-distance)
  - with the L_2-norm using gaussian noise is ``optimal''

How does it work?

- Given: testing sample x and noise variance σ
  - Sampling: Run $f(x+\delta)$ sufficient number of times where $\delta \sim N(0, \sigma \boldsymbol{I})$ and compute the most frequent class
    - » 100 000 evaluations are usually enough (takes around 150 sec on imagenet)
  - The most frequent class is the final (robust) prediction (p_1 and p_2 can also be computed and hence the radius R)
  - if p_1 and p_2 are too close, then don't provide certification
    - » sampling has some error

- Notice: due to sampling, the ultimate guarantee (certificate) is **probabilistic**!

- each testing sample can have different radius R

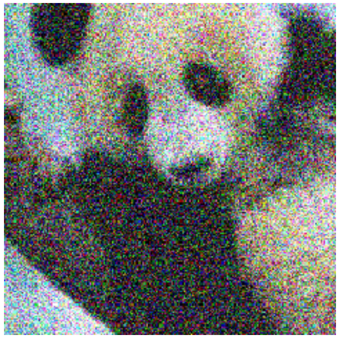
How large is the noise?

- ***the larger noise (σ) the larger R is, and hence we have stronger guarantee***

$\sigma = 0$

$R = 0$

$\sigma = 0.25$

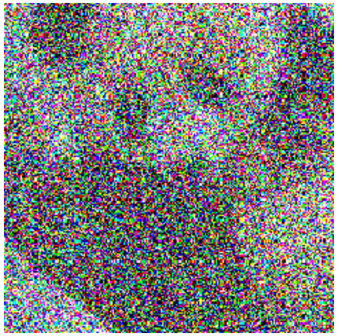


$R = 0.31$

$\sigma = 0.5$

$R = 1.025$

$\sigma = 1$



$R = 2.05$

(supposing that panda is predicted with 98% and ostrich with 2%)

Training

- In theory, the model f can be trained without noise…
- In practice, some training samples need to be noisy
  - in high dimension, the gaussian noise has no mass around its mode x and hence the noisy and non-noisy image is very different for a classifier
  - f will not learn to classify the noisy sample correctly

Corruption with Gaussian noise $\sigma = 0.5$

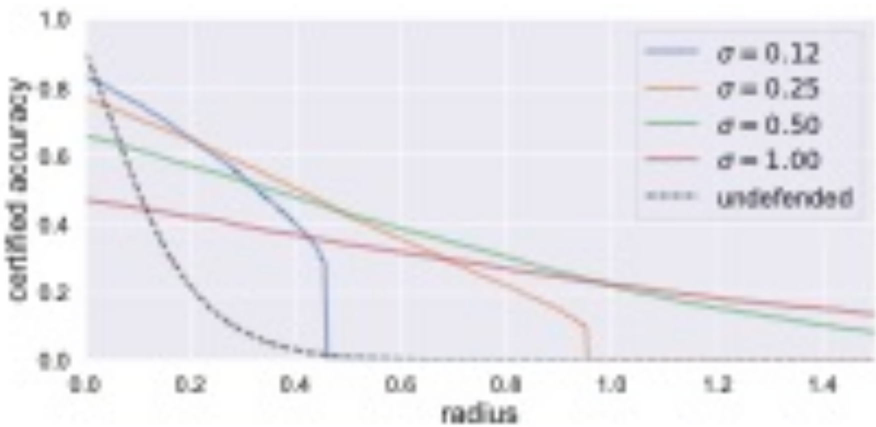
98% ostrich, 2% panda → Ostrich

Robust but incorrect

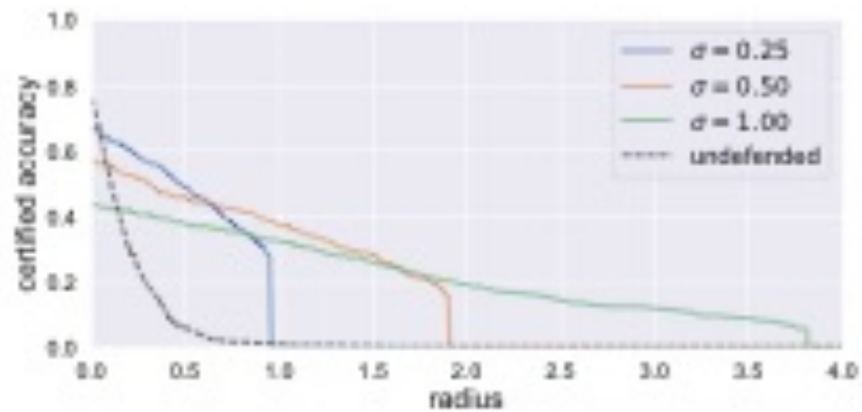
Empirical evaluation

$$\text{Certified accuracy} = \frac{\mathit{test\ sample\ x\ such\ that\ f(x) is\ correct} \wedge \text{R of x > radius}}{\mathit{all\ test\ samples}}$$

CIFAR-10



ImageNet



- Verification/certification time (per sample): 15-150 sec
- Trade-off between σ and accuracy: when σ (the noise) is high, the standard accuracy is lower (but the classifier is more robust)

DIFFERENTIAL PRIVACY AND ROBUSTNESS

Differential Privacy and Certified Robustness

- If pixels correspond to records, then differential privacy (DP) expresses the « stability » of the prediction with respect to pixels
  - Sanitizer A is $(\varepsilon, \gamma) - DP$ if

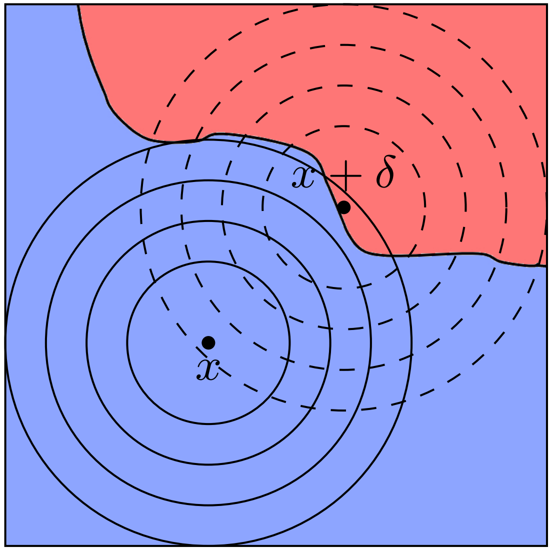
    $$\Pr(A(D) = O) \leq e^{\varepsilon} \Pr(A(D') = O) + \gamma$$

    for any neighboring dataset D and D' and output O

  - Now: $(\varepsilon, \gamma) - pixelDP$ if

    $$\mathrm{Exp}\big(A(x)\big) \leq e^{\varepsilon} \, \mathrm{Exp}\big(A(x+\delta)\big) + \gamma$$

    for any $\delta \in B_p(R)$



CONCLUSIONS

Conclusions

- Guaranteeing provable robustness is crucial in safety critical applications
- Verification of robustness on a generically trained neural network is hard
  - there are approximations with their own limitations
- Randomized smoothing perturbs training and testing to provide provable robust guarantees
  - provides robustness guarantee with arbitrarily large confidence (at the cost of computation time)
  - Pro: general approach, works for any machine learning model!
  - Con: accuracy loss can be substantial depending on the model and data

References

- Jeremy Cohen, Elan Rosenfeld, J. Zico Kolter, Certified Adversarial Robustness via Randomized Smoothing http://github.com/locuslab/smoothing
- Lecuyer et al., Certified Robustness to Adversarial Examples with Differential Privacy, IEEE S&P 2018
- Katz et al, Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks, CAV 2017
- Gehr et. al, AI^2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation, IEEE S&P 2018
- Wong et al., Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope, PMLR 2018
- Weng et al., Towards Fast Computation of Certified Robustness for ReLU Networks, PMLR 2018