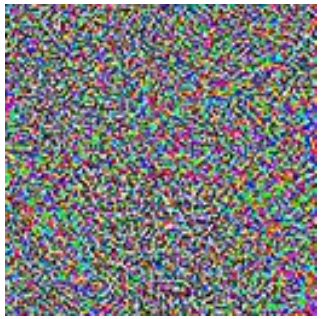
Defense Against Adversarial Attacks (Theoretic-2)

Neural Networks are Vulnerable to Adversarial Attacks

- *W.l.o.g, consider image classification problem*
- Given an image as input, a neural network (NN) model predicts a class label as the output
- Can usually craft adversarial input:
  - Indistinguishable from original input
  - Can fool NN to make wrong prediction

Predicted as
“cat”

+ 0.001 *



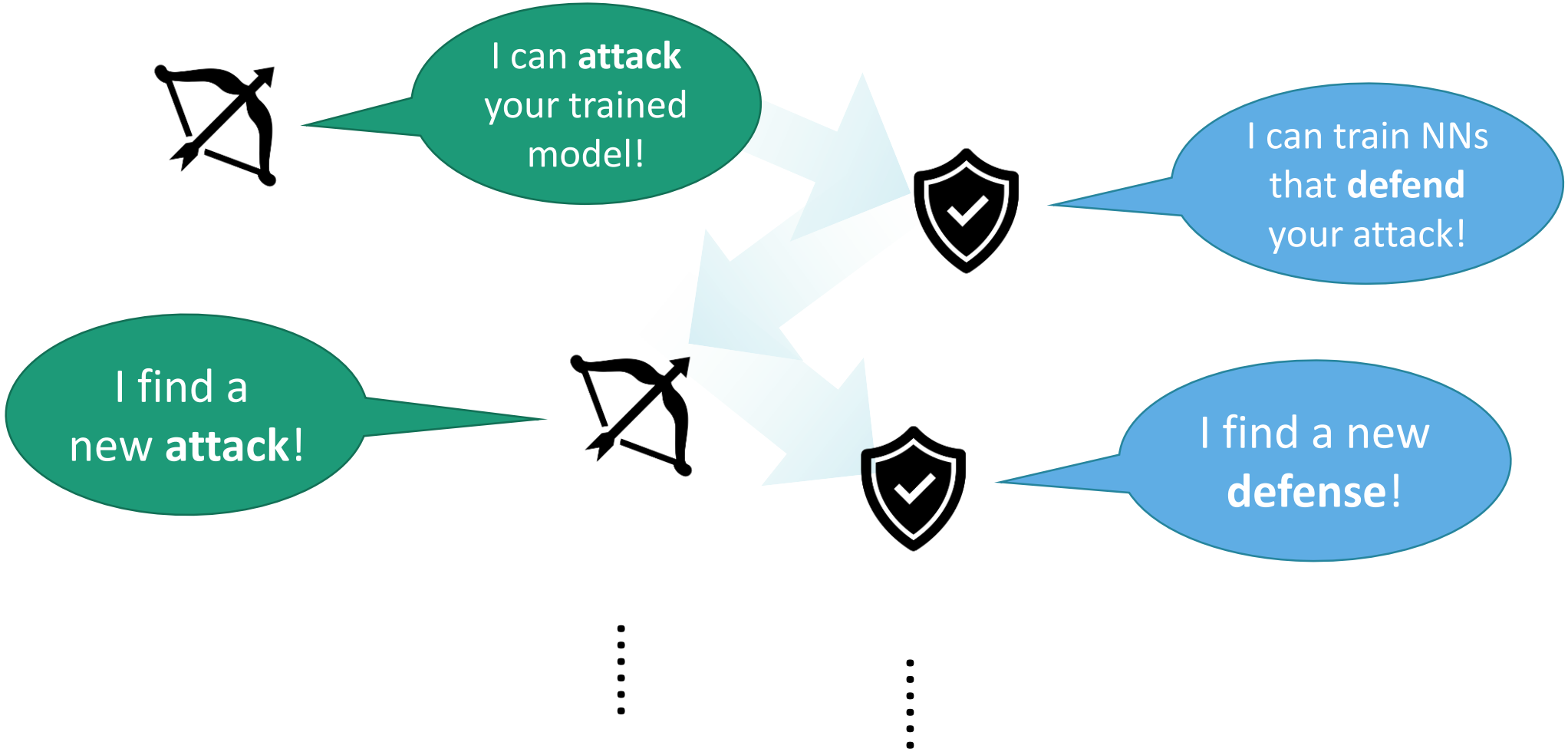
Small
Perturbation

=

Predicted as
“dog”

Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199.

Arm Race between Attacks and Defenses



End This Arm Race? Certified Robustness!

- **Certified robustness = robust training + robustness verification**
- Robust training:
  - Train NNs in a particular way
- Robustness verification:
  - For given NN model
  - Verify whether there is adversarial example for given input

➢If there is no adversarial example, any (future) attacks are destined to fail ☺

SOTA on Certified Robustness? On MNIST

- On MNIST
  - ℓ_∞ norm, $r = 0.3$
  - SOTA Certified Robust Accuracy: **93.09%**
    - *Towards Certifying ℓ_∞ Robustness using Neural Networks with ℓ_∞-dist Neurons*
    - *ArXiv: 2102.05363*
  - SOTA Empirical Robust Accuracy (against existing attacks): **96.34%**
    - https://github.com/MadryLab/mnist_challenge
    - *Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples*
    - *ArXiv: 2010.03593*

➢Not much difference

SOTA on Certified Robustness? On CIFAR-10

- On CIFAR-10
  - ℓ_∞ norm, $r = 8/255$
  - SOTA Certified Robust Accuracy: **39.88%**
    - *Fast and Stable Interval Bounds Propagation for Training Verifiably Robust Models. ArXiv: 1906.00628*
  - SOTA Empirical Robust Accuracy (against existing attacks): **65.87%**
    - Leaderboard: https://robustbench.github.io/
  - ℓ_∞ norm, $r = 2/255$
  - SOTA Certified Robust Accuracy: **68.2%**
    - *Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers. NeurIPS 2019*

➢Still a gap

SOTA on Certified Robustness? On ImageNet

On ImageNet

- ℓ_2 norm, $r = 2.0$
- SOTA Certified Robust Accuracy: 27%
  - *Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers. NeurIPS 2019*

➢Still hard (also for empirical robustness)

- We maintain the SOTA results @ https://github.com/AI-secure/Provable-Training-and-Verification-Approaches-Towards-Robust-Neural-Networks

Preliminaries

Towards Precise Definition

We need precise definition for certified robustness

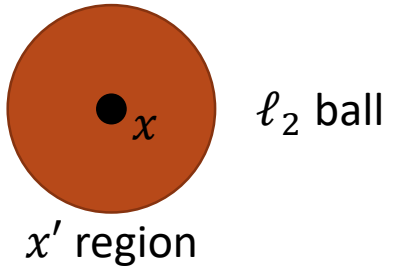
- Given an image x with d dimension,
- with ground-truth label $y \in \{1,2,\dots,C\}$ (denoted as $y \in [C]$)
- The model $F(\cdot)$ gives its prediction

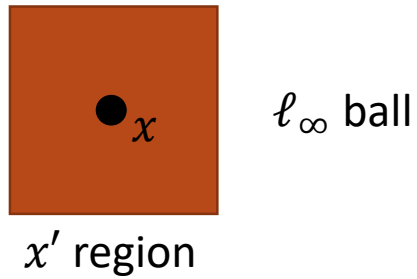
- We not only require $F(x) = y$
- but also for any x' *close to* x, $F(x') = y$

Towards Precise Definition (Cont.d)

- How to characterize "close"?
  - Usually characterize by ℓ_p distance $\leq$ pre-defined threshold ϵ

  - Typically choose ℓ_2 or ℓ_∞
  - ℓ_2: require $|| x' - x ||_2 \leq \epsilon$
  - ℓ_∞: require $|| x' - x ||_\infty \leq \epsilon$





Robust Accuracy

- We usually evaluate **certified robustness** by **robust accuracy**

Under given ℓ_p and budget ϵ,

- Robust accuracy of F: $\dfrac{\#\text{ of certifiably robust samples in dataset}}{\#\text{ of samples in dataset}}$

Hardness of Achieving Certified Robustness

- We need to train neural networks, which is
  - *Not only* good given normal inputs
  - *But also* robust against perturbed inputs
  - *And* we can verify its robustness against **any** perturbed inputs

Goal of standard training

Goal of empirical defense

Goal of certified defense

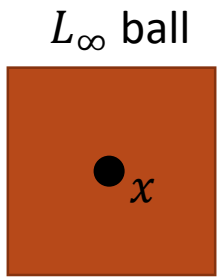
- Unfortunately, verifying the robustness of NN is NP-Complete
  - Current exact verification approaches only work when model has <1,000 neurons
  - Neural networks usually have 10^3 – 10^6 neurons

Weng, Lily, et al. "Towards Fast Computation of Certified Robustness for ReLU Networks." ICML. 2018.

Incomplete Robustness Verification

- Since exact robustness verification is NP-Complete
- we often turn to **incomplete robustness verification**
  - They use over-approximated region

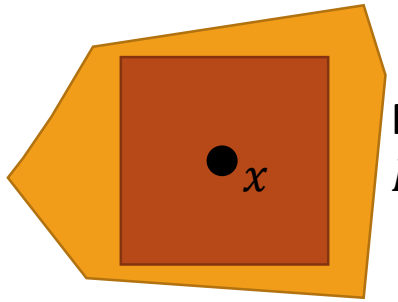
Complete Verification:



For all x' in **brown** region, $F(x') = F(x) = y_0$?

→ **NP-Complete Problem**

Incomplete Verification:

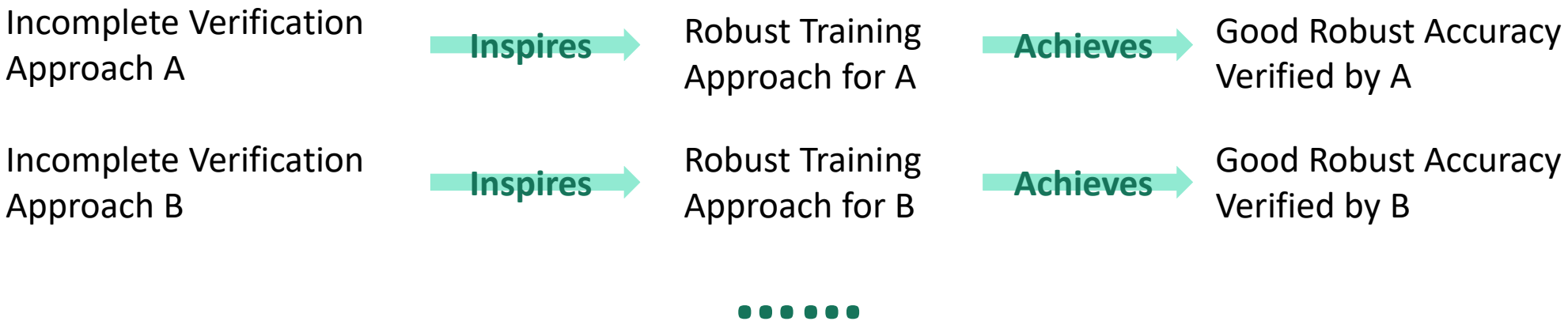


For all x' in **light brown** region, $F(x') = F(x) = y_0$?

→ **Have Efficient Algorithm**

Incomplete Verification + Training

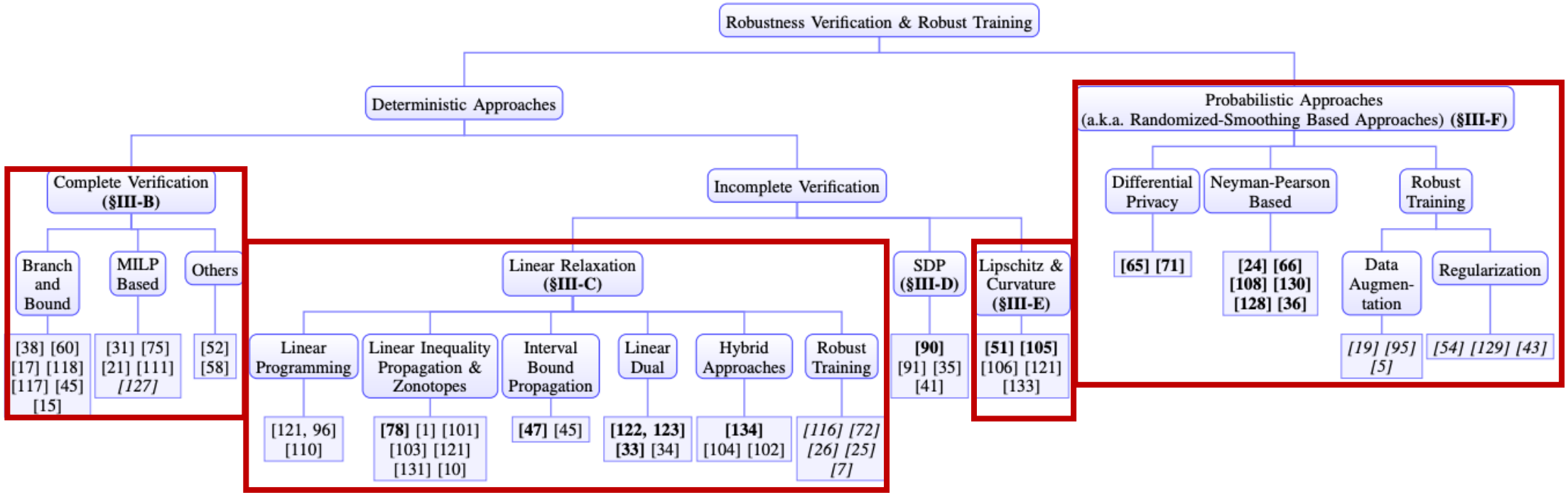
- To provide certified robustness, now we need to
  - Determine an incomplete robustness verification approach
  - Train the neural network to be robust in **over-approximated region**
- Currently, the common practice is



Note: Robust training for B usually cannot achieve good robust accuracy if verified by A

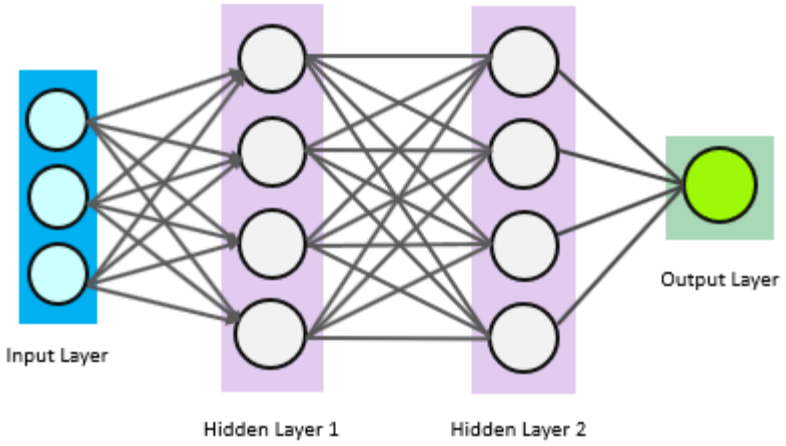
Main Taxonomy

- Since verification inspires training approach, we categorize certified robustness approaches by verification methods



Typical Approaches for Certified Robustness

Background: Neural Network Structure



- **Input layer**: vector x_0
- **Weights**: $(W_0, b_0), (W_1, b_1), \ldots (W_{L-1}, b_{L-1})$.
- **Activation function**:
  - $\mathrm{ReLU}(x) = \max\{x, 0\}$
- **Computation**:
  - $x_1 = \mathrm{ReLU}(W_0 x_0 + b_0)$,
  - $x_2 = \mathrm{ReLU}(W_1 x_1 + b_1)$,
  - …
  - $x_L = W_{L-1} x_{L-1} + b_{L-1}$
- **Output**: x_L - confidence score for each class

https://www.educba.com/deep-learning-technique/

Complete Verification

- NN = linear operations + ReLU
- Robustness verification = an optimization problem

Problem 1 (Robustness Verification as Optimization). *Given a neural network* $f_\theta : \mathcal{X} \mapsto \mathbb{R}^C$, *input instance* $x_0 \in \mathcal{X}$, *ground-truth label* $y_0 \in [C]$, *target label* $y_t \in [C]$ *and the radius* $\epsilon > 0$, *we define the following optimization problem:*

$$\mathcal{M}(y_0, y_t) = \underset{x}{\text{minimize}}\ f_\theta(x)_{y_0} - f_\theta(x)_{y_t} \ \ s.t.\ \ x \in B_{p,\epsilon}(x_0). \tag{2}$$

One can certify that f_θ *is robust at* x_0 *within radius* ϵ *w.r.t.* $\mathcal{L}_p$ *norm, as long as* $\mathcal{M}(y_0, y_t) \geq 0, \forall y_t \in [C]$.

- What we need to do?
- Express ReLU by easier operators, then let optimizer handle it ☺

https://www.educba.com/deep-learning-technique/

ReLU Formulation

- MILP Problem:
  - Linear programming + some variables being binary
  - Express ReLU:
    - If we know $x \in [l, u]$,
    - $y = \text{ReLU(x)} = \max\{x, 0\} \Leftrightarrow$

$$(y \leq x - l(1 - a)) \wedge (y \geq x) \wedge (y \leq u \cdot a) \wedge (y \geq 0) \wedge (a \in \{0, 1\})$$

  - Use linear relaxation (to be introduced later) to compute l and u
  - Then, Gurobi MILP solver can handle this

Tjeng, Vincent, Kai Y. Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming." ICLR 2019

ReLU Formulation

- SMT Formulation
  - $y = \mathrm{ReLU}(x) = \max\{x, 0\} \Leftrightarrow$
  - $\big((x \leq 0) \wedge (y = 0)\big) \vee \big((x > 0) \wedge (y = x)\big)$

  - Then, SMT solver can handle this

Branch-and-Bound

- A strategy inspired from above formulation
- Conditioned on two branches: $x \leq 0$ and $x > 0$
  - Reduce to linear constraints: $y = 0$ or $y = x$

  - Select some neurons to condition on => reduced to linear constraints
  - Relax other neurons by linear constraints
    - Recall the MILP formulation:

$$(y \leq x - l(1-a)) \wedge (y \geq x) \wedge (y \leq u \cdot a) \wedge (y \geq 0) \wedge (a \in \{0,1\})$$

$a \in [0,1]$

  - Solve the LP problem:
    - Though still slow, but in polynomial time

Wang, Shiqi, et al. "Formal security analysis of neural networks using symbolic intervals." USENIX Security 18

Actual Performance

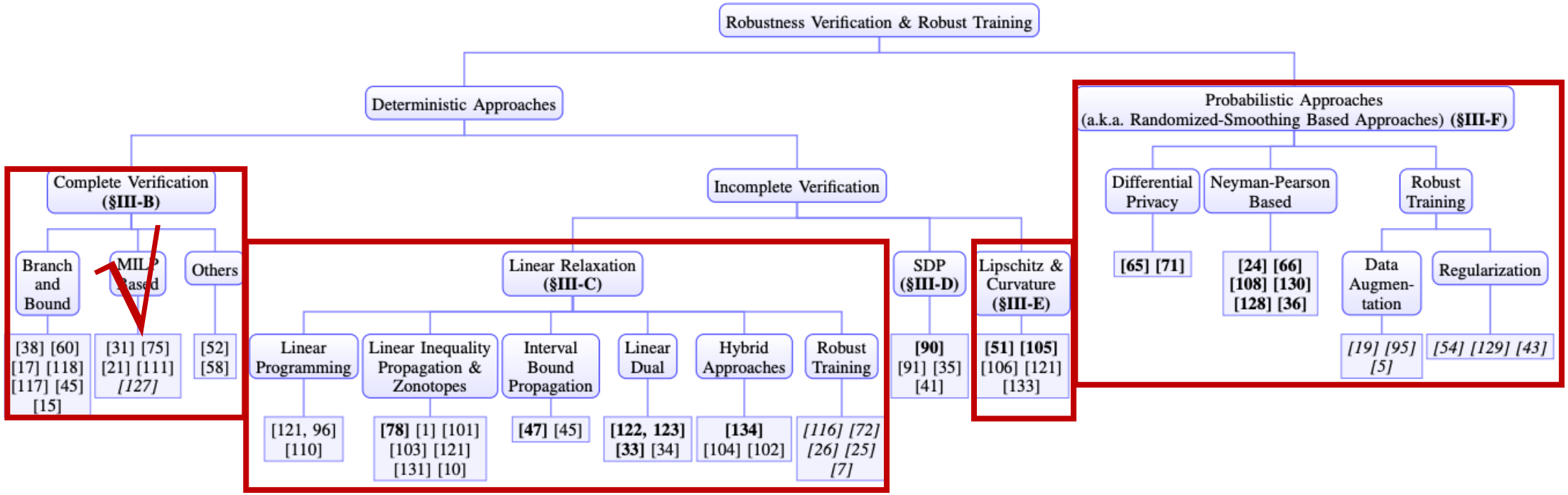
- MILP faster or similar to Branch-and-Bound,
- Branch-and-bound faster than SMT

- For **non-certifibly-trained NNs,** they all can handle only **<1k** neurons
  - Small models on MNIST, FashionMNIST
- For **certifiably-trained NNs**, MILP can handle with **10k** neurons
  - Medium models on MNIST, FashionMNIST
  - Small models on CIFAR-10

Outlook

- Complete verification cannot escape the NP-Complete barrier
  - Their worst-case complexity is always exponential unless NP=P
  - The trend is to add different heuristics
    - Towards good performance in practice
    - Maybe hard to have significant improvement

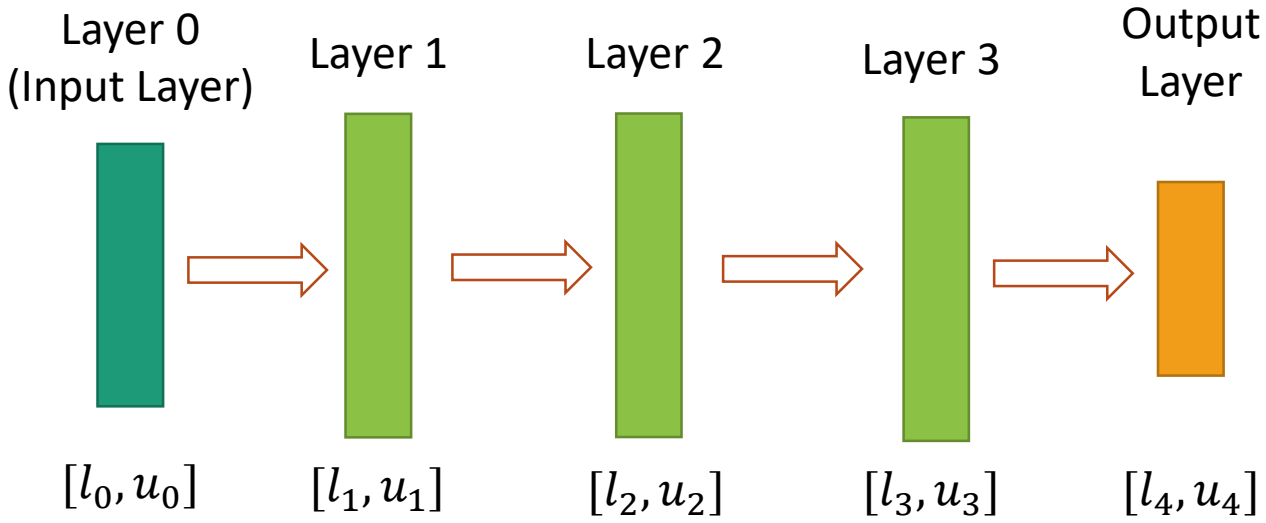
Main Taxonomy



Interval Bound Propagation

- **Key idea: Propagating interval of output through layers.**
- *Base Case:* Given input x, perturbed input range is $[x - \epsilon, x + \epsilon]$
  - ϵ: predefined radius under L_∞ norm
- *Induction:*
  - At layer i, suppose the range is $x_i \in [l_i, u_i]$
  - Then, $W_i x_i + b_i$ is in range: $[W_i^- u_i + W_i^+ l_i + b_i, W_i^+ u_i + W_i^- u_i + b_i]$.
    - $W_i^- = \min\{W_i, 0\}, W_i^+ = \max\{W_i, 0\}$ elementwise
  - After ReLU, we get range for layer $i + 1$:
    $[\text{ReLU}(W_i^- u_i + W_i^+ l_i + b_i), \text{ReLU}(W_i^+ u_i + W_i^- u_i + b_i)]$.

Interval Bound Propagation (Cont. d)



- We can get the output interval of the last layer: $[l_L, u_L]$,
  - Let y_0 be the true class, y' be an arbitrary another class
  - If $(l_L)_{y_0} \geq (u_L)_{y'}$, for any perturbed input, we won't predict the class y'
  - So, the model is robust

Interval Bound Propagation (Cont. d)

- As we can see, the interval bound propagation is a loose verification approach.
  - For example, $x \in [1, 2], y \in [0, 1]$.
  - Apparently, $x + y \geq x - y$.
  - But because $x + y \in [1, 3], x - y \in [0, 2]$.
  - $[1, 3]$ and $[0, 2]$ overlap, we cannot verify anything…

Robust Training for Interval Bound Propagation

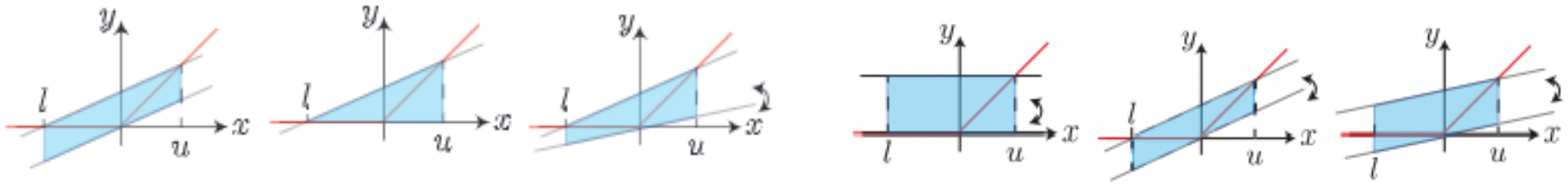
- How to solve? – Robust Training for Interval Bound Propagation
  - Train NN to be easy to be verified
  - Use **lower bound** for true class, and **upper bound** for any other classes in the training objective
  - Train NN to have small loss with these (loose) bounds
  - *Similar to PGD adversarial training, but with true bounds*

- Advantage: verification is easy and very fast.
- Drawback: hard to train
  - Require some tricks: warm start, mix with standard loss, …

Gowal, Sven, et al. "Scalable verified training for provably robust image classification." ICCV 2019

Linear Inequality Propagation

- Can we have tighter verification approach?
- Instead of propagating intervals, we propagate linear bounds
- Key Ingredient: Linear Relaxation for ReLU

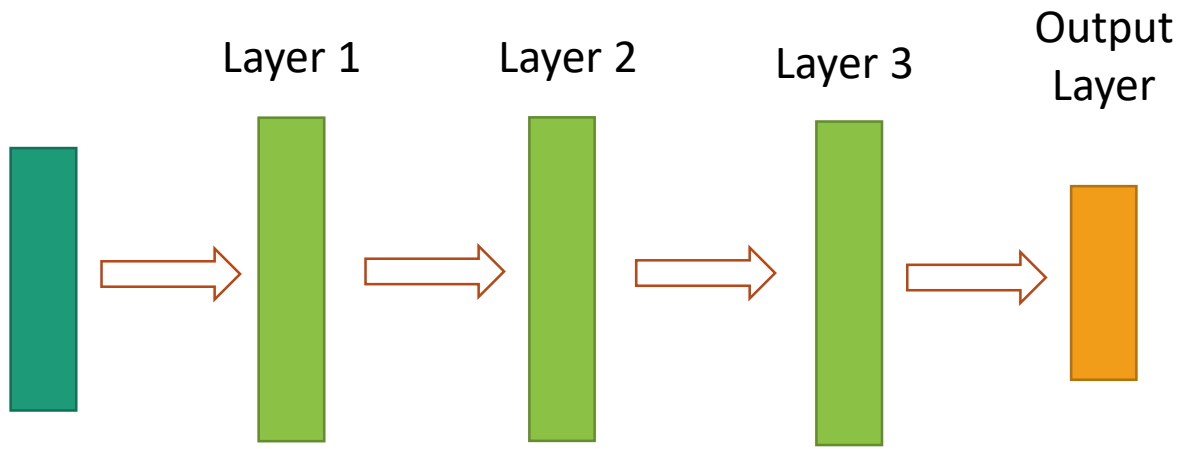


Different Linear Relaxations for ReLU (ReLU(x) = max {x, 0})

Weng, Lily, et al. "Towards fast computation of certified robustness for relu networks." ICML 2018
Wong, Eric, and Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope." ICML 2018
Singh, Gagandeep, et al. "Fast and Effective Robustness Certification." NeurIPS 2018

Linear Inequality Propagation (Cont.d)

- Propagate linear bounds:



Interval Bound Propagation	$[l_0, u_0]$	$[l_1, u_1]$	$[l_2, u_2]$	$[l_3, u_3]$	$[l_4, u_4]$
Linear Bound Propagation	$[x, x]$	$[L_1 x + l_1, U_1 x + u_1]$	$[L_2 x + l_2, U_2 x + u_2]$	$[L_3 x + l_3, U_3 x + u_3]$	$[L_4 x + l_4, U_4 x + u_4]$

Math Details

- Suppose we have bound for z_k:

$$\mathbf{L}_k x + b_{L,k} \leq z_k(x) \leq \mathbf{U}_k x + b_{U,k}$$

From $\hat{z}_{k+1} = \mathbf{W}_k z_k + b_k$, deduct bound for $\hat{z}_{k+1}$:

$$\begin{aligned} &(\mathbf{W}_k^+ \mathbf{L}_k + \mathbf{W}_k^- \mathbf{U}_k)x + \mathbf{W}_k^+ b_{L,k} + \mathbf{W}_k^- b_{U,k} + b_k \\ \leq &\hat{z}_{k+1}(x) \\ \leq &(\mathbf{W}_k^+ \mathbf{U}_k + \mathbf{W}_k^- \mathbf{L}_k)x + \mathbf{W}_k^+ b_{U,k} + \mathbf{W}_k^- b_{L,k} + b_k. \end{aligned}$$

Then, for next layer $z_{k+1} = \text{ReLU}(\hat{z}_{k+1})$:

apply the linear bound for ReLU

Linear Inequality Propagation (Cont.d)

- **Advantage**: tighter verification than interval bound propagation.
- **Drawback**: much slower than interval bound propagation.
  - From $O(w^2)$ to $O(w^3)$ per layer, where w is the layer width.
- Robust training:

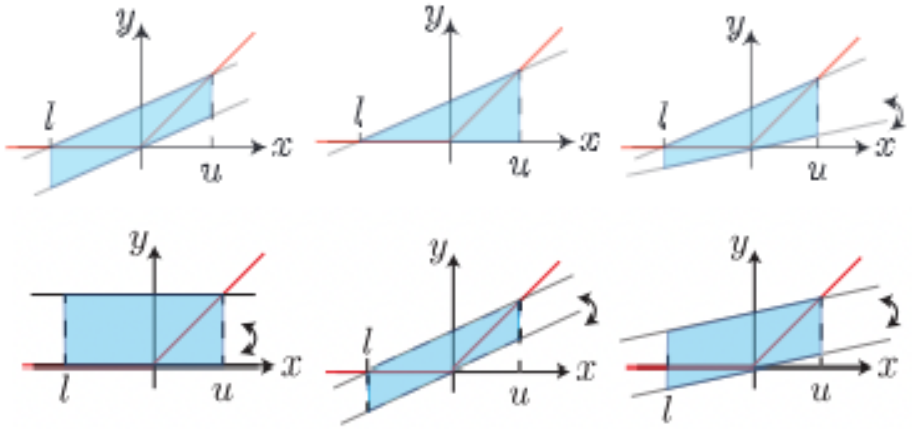
$$(\mathbf{W}_k^+ \mathbf{L}_k + \mathbf{W}_k^- \mathbf{U}_k)x + \mathbf{W}_k^+ b_{L,k} + \mathbf{W}_k^- b_{U,k} + b_k$$
$$\leq \hat{z}_{k+1}(x)$$
$$\leq (\mathbf{W}_k^+ \mathbf{U}_k + \mathbf{W}_k^- \mathbf{L}_k)x + \mathbf{W}_k^+ b_{U,k} + \mathbf{W}_k^- b_{L,k} + b_k.$$

  - Can use the lower linear bound and upper linear bound in the training objective
  - Similar to interval bound propagation.
  - Easy to train, but slow (i.e., not scalable).

Still Tighter?

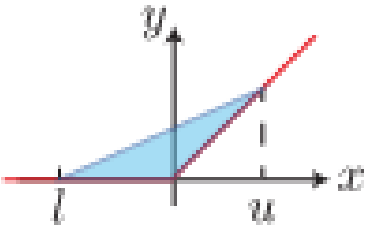
- Use linear constraints, instead of linear bounds
- Linear Bounds:

One bottom line, one upper line



- Linear Constraints:

Two bottom lines, one upper line



- Drawback: cannot propagate the linear bounds
- Now need to solve expensive LP problem

Theoretical Discussions on LP

- Salman et al. (NeurIPS 2019)
  - We tried the tightest LP problem
  - We prove that we are the tightest among all linear relaxations
  - We spent several weeks to solve it, on 100 CPUs
  - the verification result is still bad…
  - So, there is a **convex barrier**
- Their tightest LP problem:

$$\min_{(x^{[L+1]},z^{[L]})\in\mathcal{D}} c^\top x^{(L)} + c_0 \quad \text{s.t.} \quad z^{(l)} = \mathbf{W}^{(l)}x^{(l)} + b^{(l)}, \underline{\sigma}^{(l)}(z^{(l)}) \le x^{(l+1)} \le \overline{\sigma}^{(l)}(z^{(l)}), \forall l \in [L], \quad (\mathcal{C})$$

$$\underline{\sigma}_{ReLU}(z) = \max(0, z), \quad \overline{\sigma}_{ReLU}(z) = \frac{\overline{z}}{\overline{z}-\underline{z}}(z - \underline{z}),$$

Salman, Hadi, et al. "A convex relaxation barrier to tight robustness verification of neural networks." NeurIPS 2019

Theoretical Discussions on LP (Cont.d)

- Singh et al. (NeurIPS 2019)

$$\min_{(x^{[L+1]},z^{[L]})\in\mathcal{D}} c^\top x^{(L)} + c_0 \quad \text{s.t.} \quad z^{(l)} = \mathbf{W}^{(l)}x^{(l)} + b^{(l)}, \underline{\sigma}^{(l)}(z^{(l)}) \le x^{(l+1)} \le \overline{\sigma}^{(l)}(z^{(l)}), \forall l \in [L], \quad (\mathcal{C})$$

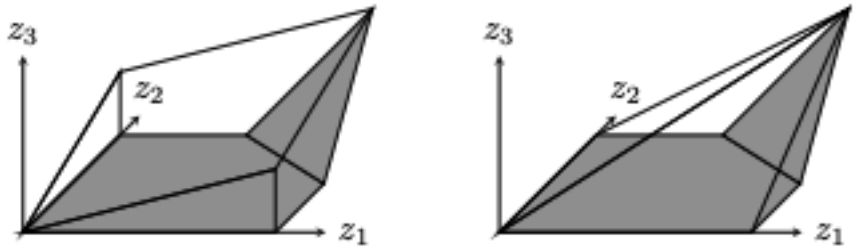
$$\underline{\sigma}_{ReLU}(z) = \max(0, z), \quad \overline{\sigma}_{ReLU}(z) = \frac{\overline{z}}{\overline{z}-\underline{z}}(z - \underline{z}),$$

  - This is not the tightest:
    - I tried to add constraints that combine some neurons together and it becomes tighter ☺
    - $\ldots \le z_i^{(l)} + z_j^{(l)} \le \cdots \to \cdots \le x_i^{(l)} + x_j^{(l)} \le \cdots$
    - $\ldots \le z_i^{(l)} - z_j^{(l)} \le \cdots \to \cdots \le x_i^{(l)} - x_j^{(l)} \le \cdots$

Singh, Gagandeep, et al. "Beyond the single neuron convex barrier for neural network certification." NeurIPS 2019.

Theoretical Discussions on LP (Cont.d)

- What happens here?



- The convex set of y after **vector** space transformation $x \mapsto y$:

$$z = Wx + b, y = \mathrm{ReLU}(z)$$

- Does **not** take the form of elementwise ReLU relaxation
- *Detail analysis require combinatoric optimization background*

Tjandraatmadja, Christian, et al. "The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification." NeurIPS2020

Interesting Fact

- Tighter relaxation does not yield better robust training…
  - Linear bound propagation worse then interval bound propagation
  - Interval bound propagation may worse than hybrid
    - But just a tiny gap
  - **Why?**
  - **Conjecture**: tighter relaxation induces rigid landscape
    - Hard to optimize
    - *Certified Defenses: Why Tighter Relaxations May Hurt Training? arXiv:2102.06700.*

Outlook

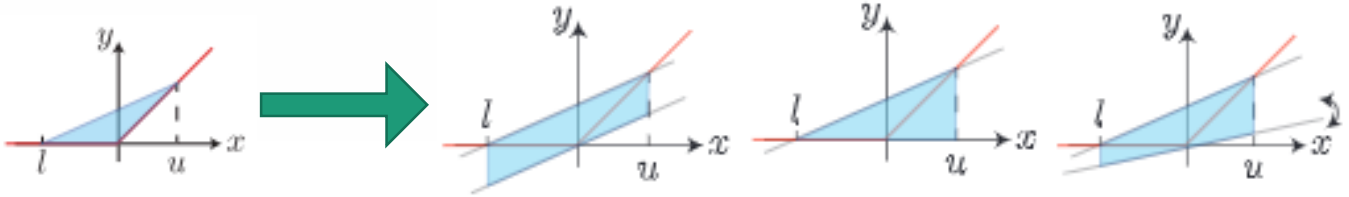
- The linear relaxation approaches are well-developed
- Some recent progress:
  - **More universal toolkits**
    - Auto-linear propagation of different forms of NN, like Autograd for DL training
      - *E.g., LiRPA (NeurIPS2020)*
  - **More theoretical analysis**
    - Analysis of the tightest convex relaxations
      - *E.g., Tjandraatmadja, Christian et al (NeurIPS 2020), Alessandro De Palma et al (ICLR 2021)*
  - **Study of training mechanisms**
    - *Does tight bound always good for training?*
      - *E.g., Certified Defenses: Why Tighter Relaxations May Hurt Training?*

Recall:

Branch-and-Bound for Complete Verification

- Branch-and-Bound:
  - Select some neurons to condition on $x \leq 0$ or $x > 0$
    - ➢ reduce to linear constraints
  - Relax other neurons by **linear constraints**
  - Solve the LP problem
- We can replace the linear constraints with linear bounds
  - Drawback: looser bound
  - Advantage:

    faster via bound propagation



- Observe: the linear bound is differentiable
  - Can optimize the lower bound's slope toward tigher bound

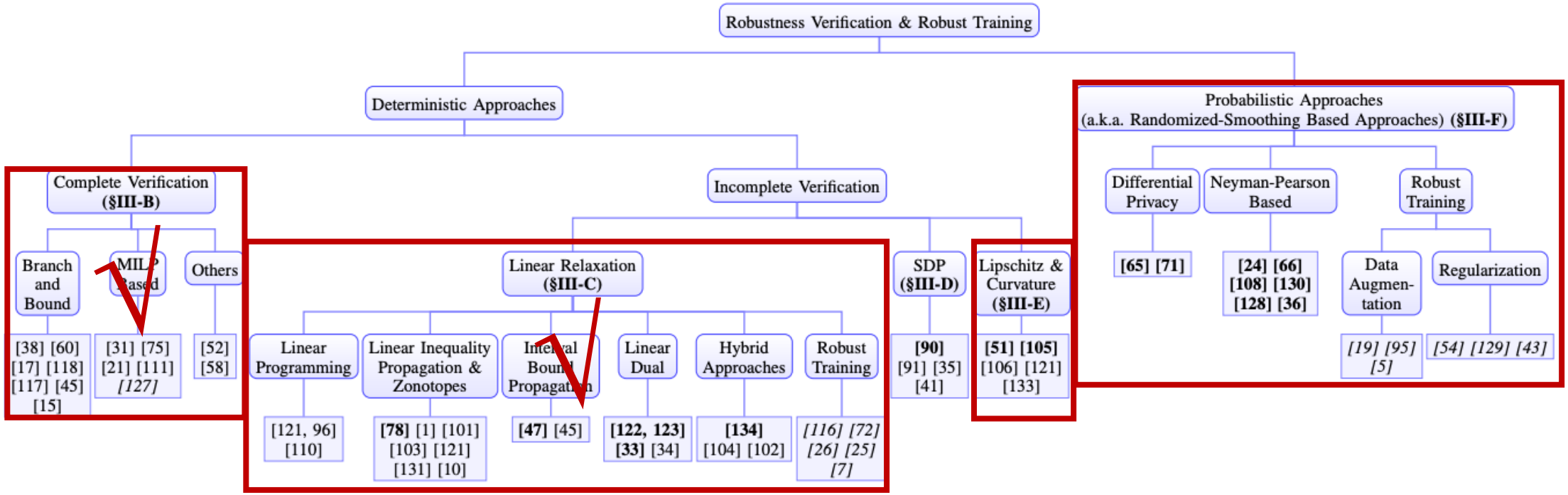
Linear Bound → Complete Verification

- Plugin the lower bound optimization module into branch-and-bound
  - Get a new complete verification approach

- This approach:
  - Looser bound on each branch (due to the use of linear bound)
  - But faster on each branch (free from LP solving)
- Be able to try more branches, and eventually be tighter and more efficient

Xu, Kaidi, et al. "Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers." ICLR2021

Main Taxonomy



Lipschitz Bound

- Lipschitz bound certification is very straightforward

$$\phi(x) = \phi_K(\phi_{K-1}(\dots\phi_1(x; W_1); W_2)\dots; W_K)\ ,$$

- ReLU has Lipschitz bound 1
- So, the whole function has Lipschitz bound $\prod_{i=1}^{K} ||W_i||$
- We can just use this bound… or tighten it

Szegedy, Christian, et al. "Intriguing properties of neural networks." ICLR 2014

Globally Lipschitz Neural Nets

- In fact, we can train NN with small global Lipschitz using regularizer
  - Achieves similar certified robustness as linear relaxations on ℓ_2
- The usage of Linfty neuron:

$$u(\boldsymbol{z}, \theta) = \|\boldsymbol{z} - \boldsymbol{w}\|_\infty + b,$$

  - Intrintically, it has Lipschitz constant 1
  - *Bohang Zhang, Zhou Lu, Tianle Cai, Di He, Liwei Wang. "Towards Certifying ℓ_∞ Robustness using Neural Networks with ℓ_∞-dist Neurons. " ArXiv: 2102.05363*

Globally Lipschitz Neural Nets (Cont.d)

- Directly train tighter global Lipschitz bound
  - How? Power methods to get the spectral norm and eigenvector
  - Then regularize it

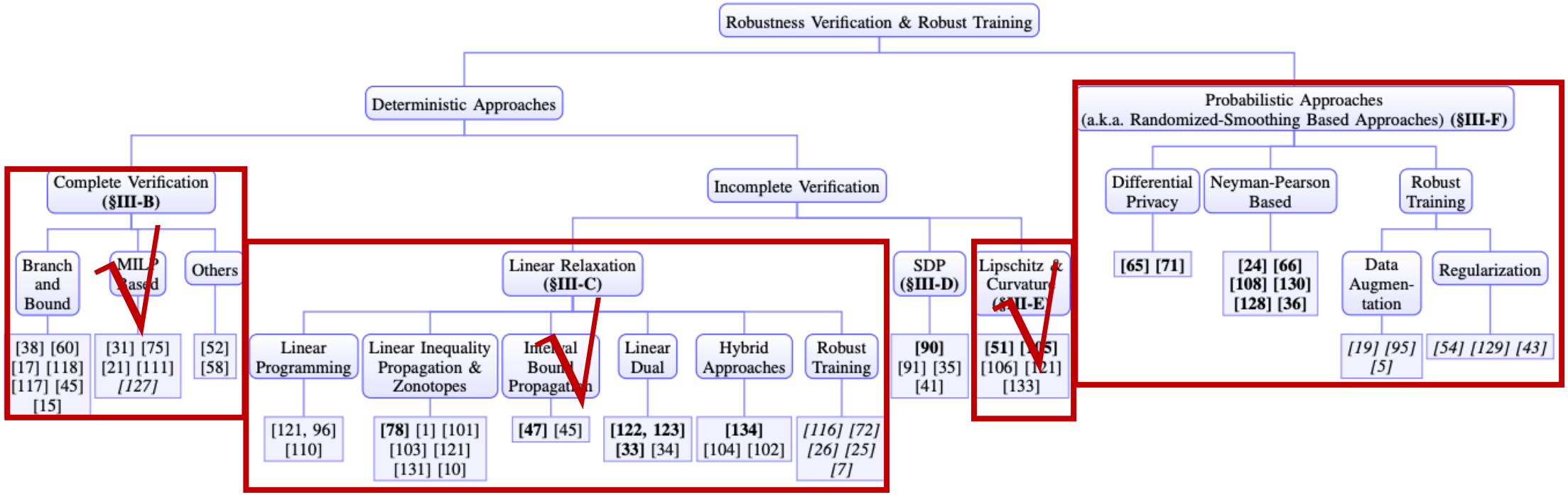
- *Tsuzuku, Yusuke, Issei Sato, and Masashi Sugiyama. "Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks." NIPS 2018.*
- *Lee, Sungyoon, Jaewook Lee, and Saerom Park. "Lipschitz-Certifiable Training with a Tight Outer Bound." NeurIPS 2020*

Outlook

- This is a novel research branch that needs further inspection
  - Might be very promising

- For example, design new model structures?

Main Taxonomy



Randomized Smoothing

Cohen, Jeremy, Elan Rosenfeld, and Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing." ICML 2019

High-Level Idea

- First, train an NN f (the "base classifier") under Gaussian data corruption:

Clean Image

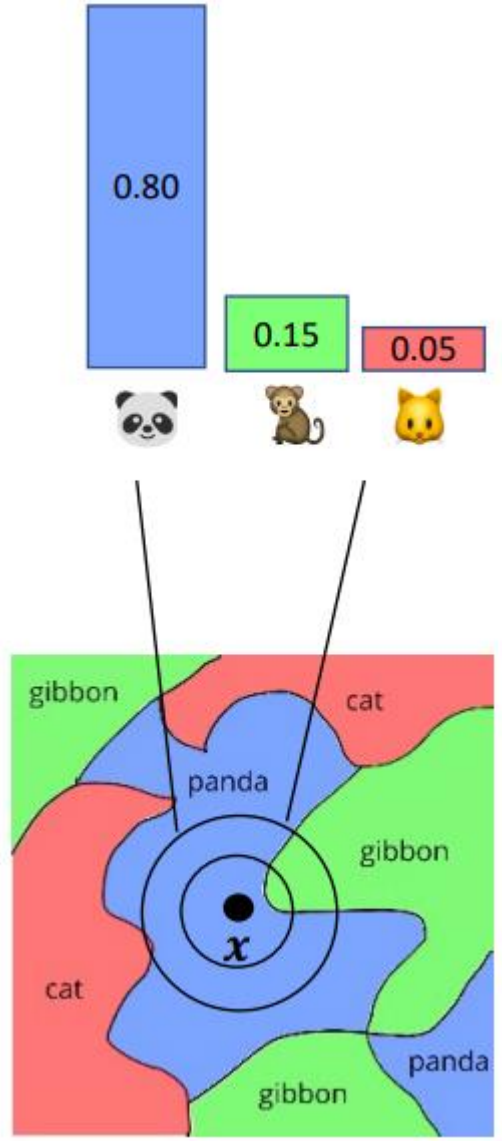
Corrupted by Gaussian Noise

- Then, smooth f into a new classifier g (the "smoothed classifier"), defined as follows:

Adapted from Cohen et al's slides

Randomized Smoothing

- $g(x)$ = the most probably prediction by f under random Gaussian corruptions of x
- Example:
  - Consider the input x with label panda.
  - Suppose that when f classifies $N(x, \sigma^2 I)$:
    - Panda is returned with probability 0.80
    - Gibbon is returned with probability 0.15
    - Cat is returned with probability 0.05
  - Then $g(x)$ = panda

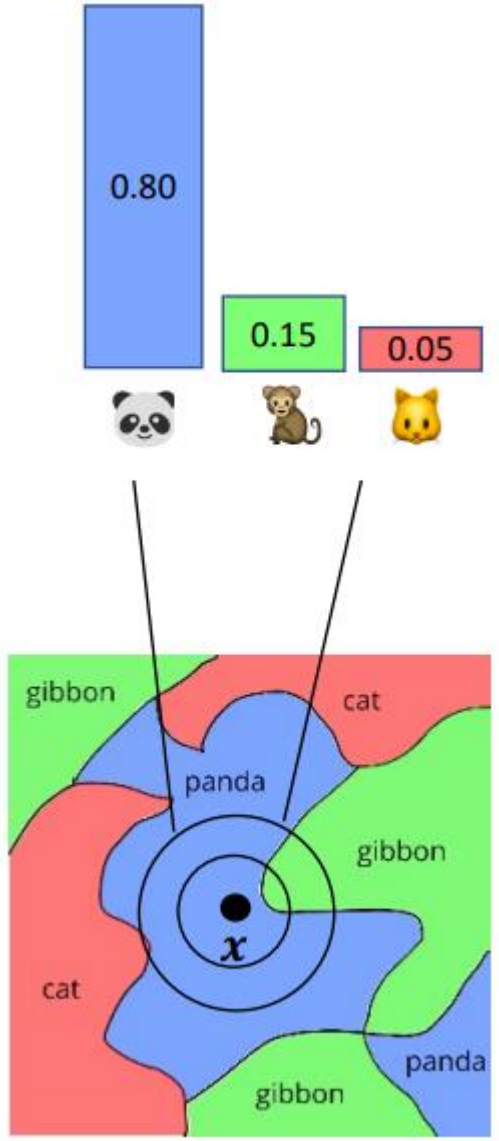


Adapted from Cohen et al's slides

Class Probabilities Change Slowly

- If we shift this Gaussian, the probabilities of each class can't change by too much

- Therefore, if we know the class probabilities at the input x, we can certify that for sufficiently small perturbations of x, the panda probability remain higher than gibbon probability



Adapted from Cohen et al's slides

Robustness Guarantee

- Let p_A be the probability of the top class (panda)
- Let p_B be the probability of the runner-up class (gibbon)
- Then g probably returns the top-class panda within an ℓ_2 ball around x of radius

$$R = \frac{\sigma}{2}\left(\Phi^{-1}(p_A) - \Phi^{-1}(p_B)\right),$$

- Where Φ^{-1} is the inverse standard Gaussian CDF

Adapted from Cohen et al's slides

Advantages of Randomized Smoothing

- To derive robustness guarantee, we don’t need to analyze the neural network structure
  - In previous approaches, we compute some information from complex math of neural networks
  - Here, we only need to know the probability of predicting each class under Gaussian noise
- So, it works for large NNs, like those for ImageNet
- Also, it does not rely on NN structure, so it is generalizable
  - Can be generalized to defend against data poisoning attack, label flipping attack, etc

Limitations of Randomized Smoothing

- Though give robustness guarantee under ℓ_2 norm
  - The guaranteed radius is small
  - Theoretically **intractable** to give good guaranteed radius **under ℓ_∞ norm**
  - *Yang, Greg, et al. "Randomized smoothing of all shapes and sizes." ICML 2020*
  - *Blum, Avrim, et al. "Random smoothing might be unable to certify l∞ robustness for high-dimensional images." JMLR 21 (2020): 1-21.*
  - *...*
- Need to train an NN with high accuracy under large noise
  - Need to sacrifice much accuracy

Circumvent the Limitation: Tighter Certification

- Indeed, we can certify larger radius for randomized smoothing
- The classical randomized smoothing only leverages P_A and P_B to derive the certification

$$\mathcal{M}_{l_{\text{targ}}} := \left\{ M : \mathbb{R}^d \to [C] \,\middle|\, m^{M}_{\text{smooth}}(\boldsymbol{x}_0) = l_{\text{targ}} \right\}.$$

$$r^{\text{W}}_{\text{targ}} := \min_{M_{\text{smooth}} : M \in \mathcal{M}_{l_{\text{targ}}}} r_{\text{true}}(\boldsymbol{x}_0; M_{\text{smooth}}).$$

  - The tightest certification suffers from tightness barrier
- We can use more information!

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & \mathbb{E}_{\varepsilon\sim\mathcal{P}}[f(\boldsymbol{\delta}+\varepsilon)] \\ \text{s.t.} \quad & \mathbb{E}_{\varepsilon\sim\mathcal{P}}[f(\varepsilon)] = P_A, \\ & \int_{\mathbb{R}^d} f_1(\varepsilon) f(\varepsilon) \mathrm{d}\varepsilon = c_1, \\ & \cdots \\ & \int_{\mathbb{R}^d} f_N(\varepsilon) f(\varepsilon) \mathrm{d}\varepsilon = c_N, \\ & 0 \le f(\varepsilon) \le 1 \quad \forall \varepsilon \sim \mathbb{R}^a. \end{aligned}$$

Additional Constraints

Tighter Certification (Cont.d)

- Higher order certification:
  - *Mohapatra, Jeet, et al. "Higher-Order Certification for Randomized Smoothing." NeurIPS 2020*
  - Sample and use the gradient magnitude information

$$\|\nabla m_{\text{smooth}}(\boldsymbol{x}_0)\|_p = \|\nabla \Pr_{\varepsilon\sim\mathcal{P}}[M(\boldsymbol{x}+\varepsilon)=y]\|_p$$

- Double sampling:
  - Recently, we try to use double sampling information

$$\mathcal{M}_{l_{\text{targ}},\hat{l}_{\text{targ}}} := \left\{ M:\mathbb{R}^d \mapsto [C] \;\middle|\; \begin{matrix} m^M_{\text{smooth}}(\boldsymbol{x}_0) = l_{\text{targ}} \\ \hat{m}^M_{\text{smooth}}(\boldsymbol{x}_0) = \hat{l}_{\text{targ}} \end{matrix} \right\} \quad \text{where} \quad \hat{m}^M_{\text{smooth}}(\boldsymbol{x})_i := \Pr_{\varepsilon\sim\mathcal{Q}}[M(\boldsymbol{x}+\varepsilon)=i].$$

Overview

- Better, but not better in magnitude
- Therefore, despite that randomized smoothing is very universal, its development has been faced with a barrier…

Table 1: ℓ_2 certified robust accuracy w.r.t. different radii r's on MNIST and CIFAR-10. **DSRS** is shown in gray.

Dataset	Model	Certification Approach	Clean Accuracy	Certified Accuracy under Radius r						
				0.4	0.8	1.2	1.6	2.0	2.4	2.8
MNIST	Gaussian Augmentation	Neyman-Pearson	97.8%	93.5%	86.3%	73.4%	50.5%	25.5%	9.0%	**1.8%**
		DSRS		**93.6%**	**87.2%**	**75.3%**	**57.1%**	**34.1%**	**14.2%**	0.0%
	Consistency [Jeong and Shin, 2020]	Neyman-Pearson	97.2%	**92.8%**	86.0%	74.7%	57.2%	35.6%	13.5%	3.5%
		DSRS		**92.8%**	**86.4%**	**76.9%**	**61.5%**	**41.9%**	**23.3%**	**7.1%**
			Clean Accuracy	Certified Accuracy under Radius r						
				0.2	0.4	0.6	0.8	1.0	1.2	1.4
CIFAR-10	Gaussian Augmentation	Neyman-Pearson	70.6%	55.3%	46.2%	35.8%	24.9%	17.1%	10.5%	4.3%
		DSRS		**55.9%**	**50.3%**	**41.0%**	**30.5%**	**19.6%**	**12.9%**	**6.0%**
	Consistency [Jeong and Shin, 2020]	Neyman-Pearson	56.0%	50.5%	45.6%	41.5%	37.2%	32.3%	26.4%	19.8%
		DSRS		**50.9%**	**47.6%**	**43.9%**	**39.1%**	**34.5%**	**28.3%**	**21.7%**

Table 2: ℓ_∞ certified robust accuracy w.r.t. different radii r's on MNIST and CIFAR-10. **DSRS** is shown in gray.

Dataset	Model	Certification Approach	Clean Accuracy	Certified Accuracy under Radius r						
				2/255	4/255	6/255	8/255	10/255	12/255	14/255
MNIST	Gaussian Augmentation	Neyman-Pearson	99.1%	97.4%	95.8%	92.4%	85.2%	73.2%	50.7%	22.6%
		DSRS		**97.5%**	**96.1%**	**92.7%**	**86.8%**	**77.6%**	**60.0%**	**29.7%**
	Consistency [Jeong and Shin, 2020]	Neyman-Pearson	98.5%	**98.2%**	**96.4%**	93.9%	88.3%	78.7%	62.7%	37.8%
		DSRS		**98.2%**	**96.4%**	**94.3%**	**89.0%**	**81.9%**	**67.5%**	**43.2%**
			Clean Accuracy	Certified Accuracy under Radius r						
				1/255	2/255	3/255	4/255	5/255	6/255	7/255
CIFAR-10	Gaussian Augmentation	Neyman-Pearson	65.6%	45.3%	36.3%	26.7%	18.1%	10.9%	6.1%	1.9%
		DSRS		**45.6%**	**37.6%**	**28.8%**	**19.5%**	**13.9%**	**8.1%**	**2.1%**
	Consistency [Jeong and Shin, 2020]	Neyman-Pearson	52.6%	**45.5%**	40.6%	36.0%	30.5%	25.2%	20.3%	**15.6%**
		DSRS		**45.5%**	**40.9%**	**36.9%**	**31.9%**	**28.1%**	**22.0%**	15.2%

Comparison

- Harder datasets requires larger neural networks
  - MNIST < CIFAR10 < ImageNet
- Larger robust radius requires tighter verification approaches

Which approach is better?

	MNIST	CIFAR10	ImageNet
Small radius	Linear bound propagation	Randomized Smoothing	Randomized Smoothing
Large radius	Interval bound propagation Global Lipschitz	Global Lipschitz	Randomized Smoothing
Best Certified Robust Accuracy	~93% eps=0.3 L_∞ (almost solved)	~68% with eps=$\frac{2}{255}$ L_∞ ~39% with eps=$\frac{8}{255}$ L_∞	~43% eps=1.0 L_2

Summary

- **Young but impactful area.**
  - Emerged in 2017
  - Typical approaches proposed in 2018-2019
  - Fast-growing
- **Still large space to improve.**
  - On small dataset (MNIST), the results are satisfying
  - On large dataset (CIFAR-10, ImageNet), still need improving.
    - For example, only <40% robust accuracy on CIFAR-10 dataset

- **Possible Directions.**
  - Tighter certification
  - More robust model structure:
    - Combine NN with other more robust model;
    - AdderNet? Binary Net? …
    - Pruning?
  - Better training approach
    - Leverage diversity and transferability?
- **Beyond cetified robustness against data evasion attacks.**
  - Against data poisoning attacks
  - Against watermarks
  - Static debugging of NN
  - RL, NLP, CV
  - …