IJCAI 2021 Tutorial:

Towards Robust Deep Learning Models: Verification, Falsification, and Rectification

Wenjie Ruan[1], Xinping Yi[2], Elena Botoeva[3], Xiaowei Huang[2]

[1]University of Exeter, UK; [2]University of Liverpool, UK; [3]Imperial College, UK

Outline

Introduction

Falsification through Adversarial Attack

Rectification through Adversarial Training

Robustness Verification

Verification in Practice

Conclusions and Future Directions

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

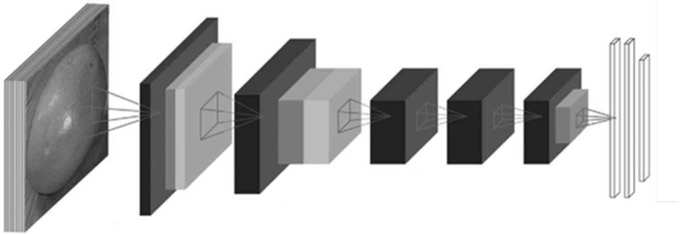
Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Introduction



Deep learning models are pervasively applied in many **safety-critical systems**!

nature reviews drug discovery

Review Article | Published: 11 April 2019

Applications of machine learning in drug discovery and development

Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran,

nature International weekly journal of science

Home | News | Research | Careers & Jobs | Current Issue | Archive | Audio & Video | For Authors

Archive > Volume 542 > Issue 7639 > Letters > Article > Article metrics > News

Article metrics for:

Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau & Sebastian Thrun

- Drug Discovery and Development
- Automatic Medical Diagnosis

- Self-driving Cars
- Autonomous Vehicles

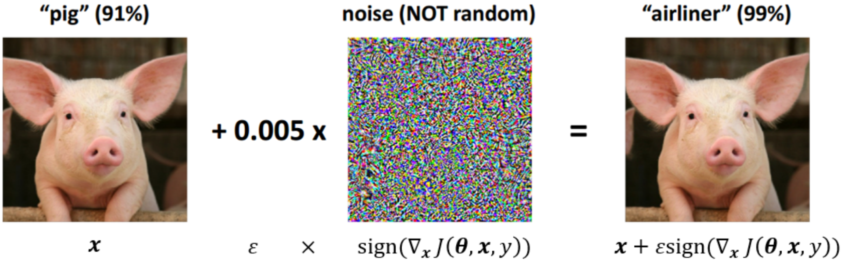
Can We Trust its Decisions?

Researchers and Practitioners may have many concerns...

- How does a deep learning model make a decision?
- Does deep learning always make a correct decision?
- Under what circumstances a deep learning model will make a wrong decision?
- ... ...

Ultimate Question: **Can we really trust the decisions made by deep learning models, especially on safety-critical applications?**

Yet we cannot **trust** deep learning models, at least **not now** ...



Simple approach to **fool** deep neural networks: Fast Gradient Sign Method (FGSM) [Goodfellow et al., 2014]

Goodfellow et al (ICLR 2014). Explaining and harnessing adversarial examples.

Vulnerability to Adversarial Examples

Such vulnerabilities are *pervasive* ...

NEWS

Home | UK | World | Business | Politics | Tech | Science | Health | Family & Education

Technology

AI image recognition fooled by single pixel change

8 hours ago | Technology

Share

MOTHERBOARD

IFICIAL INTELLIGENCE

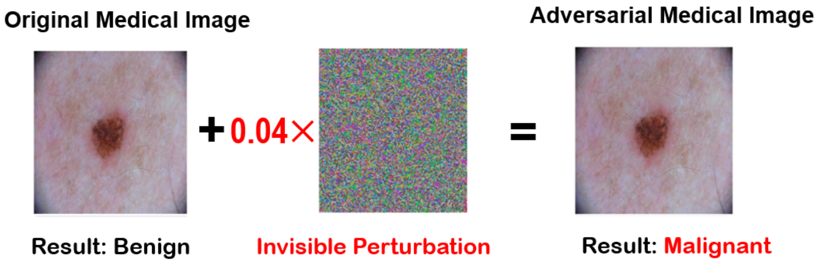
Researcher: 'We Should Be Worried' This Computer Thought a Turtle Was a Gun

Can a Machine Be Conscious?

Copyright Law Makes Artificial Intelligence Bias Worse

AI Can Be Fooled With One Misspelled Word

In Deep Medical Systems...



Adversarial Examples Against Medical Deep Learning Systems [Finlayson et al., 2019, Finlayson et al., 2018]

Finlayson, Samuel G., et al. "Adversarial attacks on medical machine learning." Science 363.6433 (2019): 1287-1289.

Vulnerability to Adversarial Examples

In Autonomous Systems...

Changing one pixel

DL Classification: Green Light

DL Classification: Red Light

Min et al. (Theoretical Computer Science, 2019), A Game-Based Approximate Verification of Deep Neural Networks with Provable Guarantees"

In Medical Record Analysis...

Original Medical Record	Record with Two Mis-spelled Words
There is extremely dense fibrous tissue in the upper outer quadrants of both breasts. This lowers the sensitivity of mammography. B.B. was placed in the region of palpable abnormality and demonstrates dense breast tissue in this region. An occasional benign-appearing calcification is present in both breasts.	There is extremely dense fibrous tissue in the upper outer quadrants of both breasts. This lowers the sensitivity of mammography. B.B. was placed in the region of palpable abnormality and demonstrates dense breast tisue in this region. An occasional benign-appearing calcifcaton is present in both breasts.
Analysis Result: Positive	**Analysis Result: Negative**

Javid et al. (ACL 2018), Hotflip: White-box adversarial examples for text classification

Fool YOLOv2 Object Detector by a real picture ...

https://www.youtube.com/watch?v=MIbFvK2S9g8

Simen et al. (CVPR Workshops 2019), Fooling automated surveillance cameras: adversarial patches to attack person detection

Adversarial Examples in Running Systems

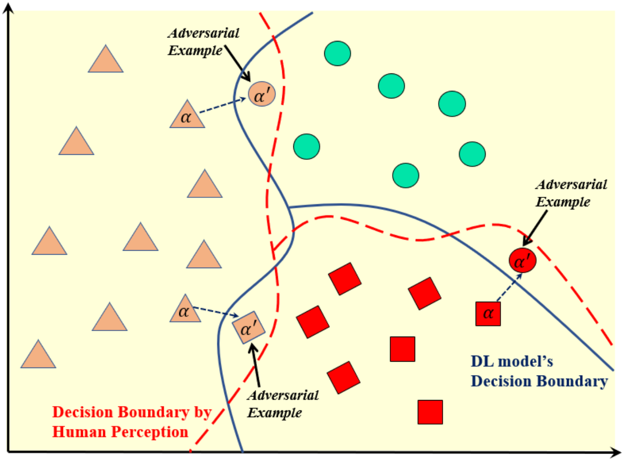
Fool an Object Classifier by a 3-D printed turtle ...
`https://www.youtube.com/watch?v=XaQu7kkQBPc`

Anish et al. (ICML 2018), Synthesizing Robust Adversarial Examples

Research Dealing with Adversarial Robustness

- Falsification (adversarial attacks, testing, etc.): How to find the weak spots of deep learning models
  $\rightarrow$ Evaluating adversarial robustness
- Rectification (adversarial defense): How to defend adversarial attacks
  $\rightarrow$ Improving the robustness w.r.t. adversarial attacks
- Verification: How to verify if a given model satisfies robustness properties for certain input constraints
  $\rightarrow$ Providing **robustness guarantees** if no counter-examples can be found

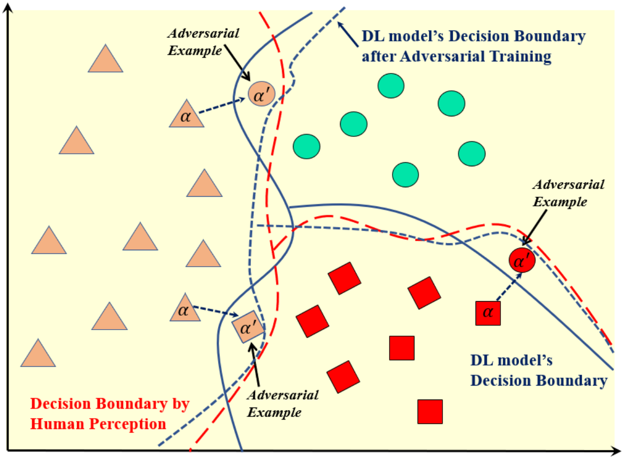
What is adversarial examples



DL model: classifies α and α' **differently**

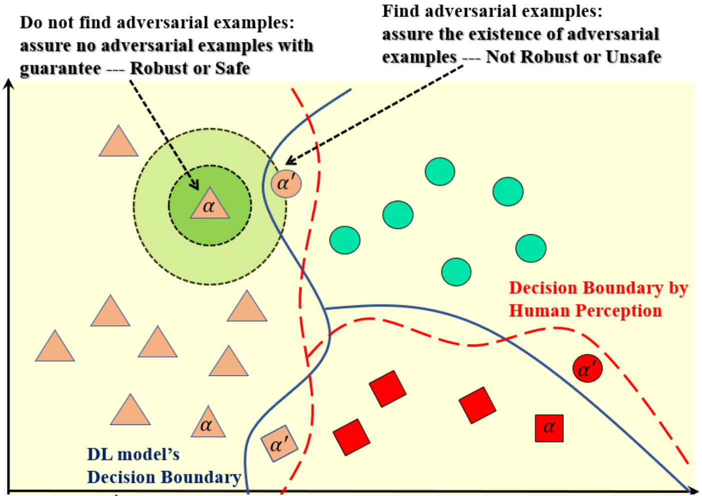
Human: should remain the **same**

An example of Defense



Injecting adversarial examples into training so the resulting DL model is **resistant** to adversarial attacks

An example of Verification



Example of (Robustness) Verification: verify if a certain input area can exclude adversarial examples with **guarantees**

University of Exeter · University of Liverpool · Imperial College London

The Aim of the Tutorial

This tutorial aims to cover a few **well-established** works from three aspects:

- Falsification via adversarial attacks
- Rectification via adversarial training
- Verification

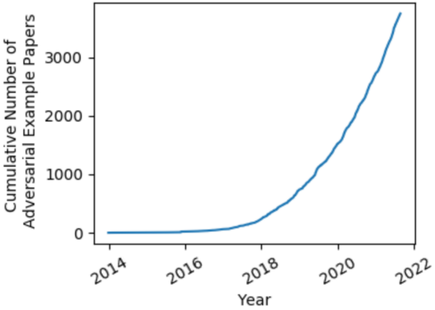


Figure: https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

Comprehensive one: **A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability, Computer Science Review. 37 (2020): 100270.**

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

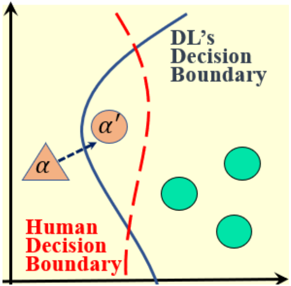
Verification in Practice

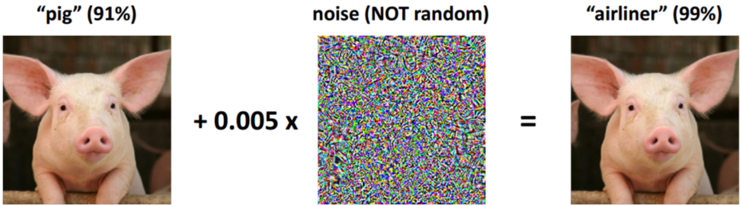
Conclusions and Future Directions

Falsification through Adversarial Attack

What is Adversarial Example

- Input: DL model f
  A correctly-classified, genuine example α
- Aim: find a perturbed example α', such that
  - f produces a different decision on α'
  - Human will produce a same decision on α and α'

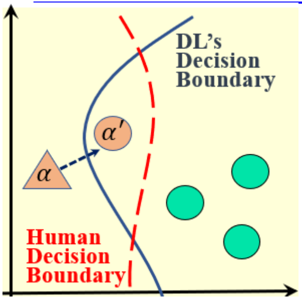




What is Adversarial Example

Problem: Given a DL model f and genuine example α, find α' such that

- $f(\alpha') \neq f(\alpha)$
- $D_{Human}(\alpha') = D_{Human}(\alpha)$



How to approximate human decision?

- Use certain distance metric to assure α' and α are small enough

How to search α' such that $f(\alpha') \neq f(\alpha)$?

- Design certain objective functions for minimization

What kind of information is required from DL model f?

- White-box v.s. Black-box

Targeted Attacks v.s. Un-targeted Attacks

- With a targeted perturbation, the attacker is able to **control** the resulting misclassification label.

- With an un-targeted perturbation, the attacker can enable the misclassification but **cannot control** its resulting misclassification label.

Distance metric to measure α' and α:

- L_p-norm distance
  - L_p-norm based attacks (e.g., $p = 0, 2, \infty$)
- Total variation of pixel displacement
  - Spatial-transformed adversarial attacks
- Metrics for measuring similarity of sentences or text
  - Attacks on NLP models

Categories of Adversarial Attacks

The information is required from DL model f (white-box or black-box):

- Hard labels only
- Confidence values
- Model's parameters and structure

The type of the model f:

- Feed-forward neural networks
- Recurrent neural networks
- Graph neural networks
- Other models

Local adversarial attack v.s. Universal adversarial attack

- Local adversarial attack:
  - find specific perturbation for **each** input
- Universal adversarial attack:
  - find a perturbation that can fool a **set** of inputs

Table of Contents

Introduction

Falsification through Adversarial Attack

- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training

- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification

- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Falsification through Adversarial Attack

Algorithms for Adversarial Attacks

One of earliest adversarial attack: optimization based formulation with L_2-norm metric

- Model $f : \mathbb{R}^{s_1} \rightarrow \{1 \ldots s_K\}$ with s_K labels
- $x \in \mathbb{R}^{s_1} = [0,1]^{s_1}$ is an input
- $t \in \{1 \ldots s_K\}$ is a target misclassification label

Find the adversarial perturbation r via

$$\begin{aligned} & \min ||r||_2 && \text{assure human-decision unchanged} \\ \textit{s.t.} \quad & \arg\max_l f_l(x+r) = t && \text{assure misclassification} \\ & x + r \in \mathbb{R}^{s_1} && \text{assure perturbed image feasible} \end{aligned} \tag{1}$$

- Solved by L-BFGS, Establish this direction

Christian et al (ICLR 2014). Intriguing properties of neural networks

FGSM Attack

Fast Gradient Sign Method is able to find adversarial perturbations with a fixed L_∞-norm constraint **very efficiently**

- θ: the model parameters,
- x, y: the input and the label
- $J(\theta, x, y)$: the loss function

Find adversarial perturbation r by linearizing the loss function around the current value of θ,

$$r = \epsilon \, \text{sign}\left(\nabla_x J(\theta, x, y)\right) \tag{2}$$

- A **one-step modification** to all pixel values to increase the loss function with a L_∞-norm constraint ϵ

Goodfellow et al (ICLR 2015). Explaining and harnessing adversarial examples

Carlini & Wagner Attack: find adversarial examples with **very small distortion**, work on L_0, L_2 and L_∞-norm

- x is an input,
- r is adversarial perturbation
- F is a designed surrogate function such as $x + r$ is able to fool the neural network when it is negative

$$\min \ \ell(r) = ||r||_p + c \cdot F(x + r) \tag{3}$$

- The optimizer Adam was directly adopt to solve this optimization problem
- The key to achieve **strong attack** is a careful design of surrogate function

---

Nicholas et al (IEEE S&P 2017). Towards evaluating the robustness of neural networks

White-box setting: full access to the target model
Black-box setting: with **limited knowledge** on the model

ZOO Attack: Model $F(x) \in [0,1]^K$ (confidence values)

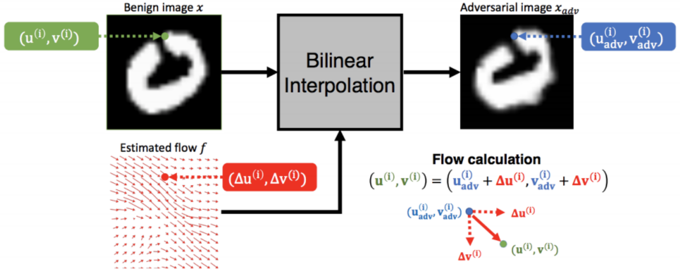
- $\min_x ||x - x_0||_2^2 + c \cdot f(x,t)$
  where $x \in [0,1]^p$ and
  $f(x,t) = max\{-\kappa, max_{i \neq t}[logF(x)]_i - [logF(x)]_t\}$
- Random coordinate gradient descent via estimated gradients by Symmetric Difference Quotient:
  $\frac{\partial g(x)}{\partial x} \approx \frac{g(x+he) - g(x-he)}{2h}$ with small h

- Nearly similar performance to white-box attack
- Key difference: only access confidence values $\rightarrow$ numerically estimate the gradient

---

Chen et al (ACM Workshop on AI&Security 2017). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models

What else can we modify? Perturb the locations of pixels



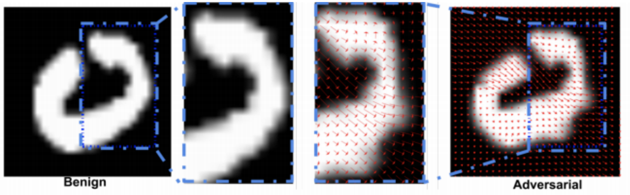
$f^* = arg\min_f \mathcal{L}_{adv}(x, f) + \tau \mathcal{L}_{flow}(f)$ minimize flow

$\mathcal{L}_{adv}(x, f) = \max(\max_{i \neq t} g(\mathbf{x}_{adv})_i - g(\mathbf{x}_{adv})_t, \kappa)$ surrogate function

Measure the pixel displacement: $\mathcal{L}_{flow}(f) = \sum_{n=1}^{pixels} \sum_{q \in \mathcal{N}(p)} \sqrt{||\Delta u^{(p)} - \Delta u^{(q)}||_2^2 + ||\Delta v^{(p)} - \Delta v^{(q)}||_2^2}$

Perform spatial transformation: $\mathbf{x}^{(i)}_{adv} = \sum_{q \in \mathcal{N}(u^{(i)}, v^{(i)})} \mathbf{x}^{(q)} (1 - |u^{(i)} - u^{(q)}|)(1 - |v^{(i)} - v^{(q)}|)$

Flow visualization on MNIST: digit "0" is misclassified as "2"

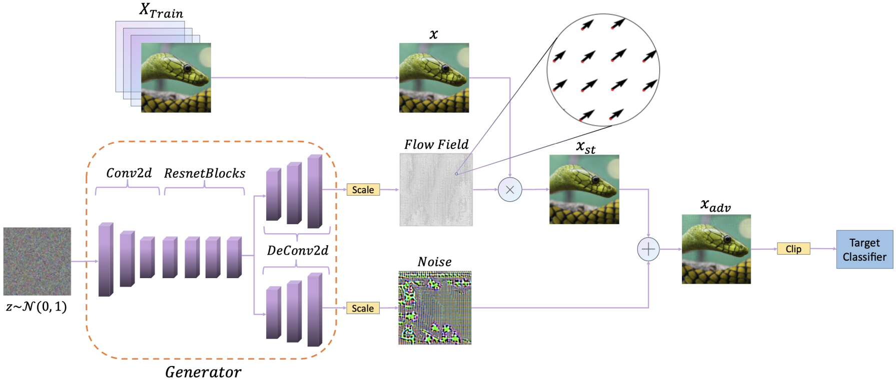


- Instead of perturbing the pixel values, adversarial attacks can be achieved by **spatial transformation**
- Different metric is required to measure pixel's **spatial** displacement

Chaowei et al (ICLR 2018). Spatially transformed adversarial examples

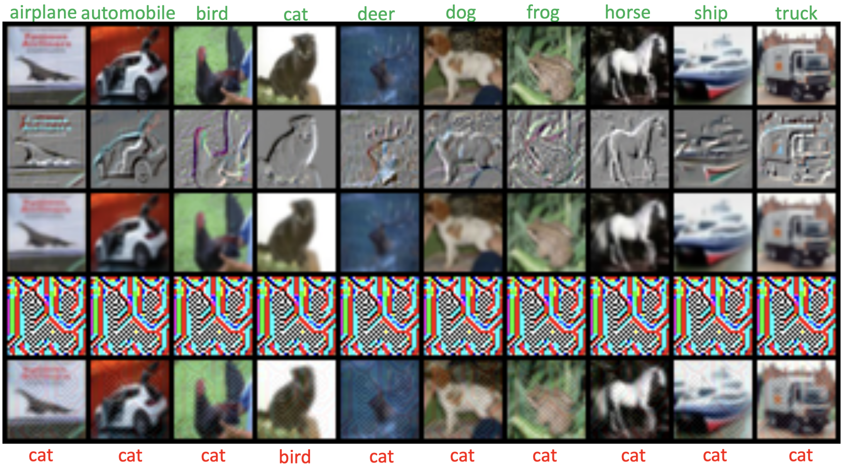
Universal Attack via Combined Perturbation

How about perturbing spatial location and pixel values simultaneously on an image set?



- **Unified solution**: L_p-norm, spatial-transformed, or both
- **Universal**: a single perturbation fools a set of input images
- **Strong transferability**: workable across unseen models in a black-box setting

Universal Attack via Combined Perturbation



Yanghao et al (ICDM 2020). Generalizing Universal Adversarial Attacks Beyond Additive Perturbations

Table of Contents

Introduction

Falsification through Adversarial Attack

- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training

- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification

- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

More Examples of Adversarial Attacks

More Attacks: on NLP models

Adversarial attack on **reading comprehension** system

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

- Adding distracting sentences (in blue)
- Prediction changes from correct one (green) to incorrect (red)

Robin et al (EMNLP 2017). Adversarial Examples for Evaluating Reading Comprehension Systems

More Attacks: on NLP models

Edit adversarial attack on sentiment analysis system:

Task: Sentiment Analysis. ***Classifier:*** Amazon AWS. ***Original label:*** 100% Negative. ***Adversarial label:*** 89% Positive.

Text: I watched this movie recently mainly because I am a Huge fan of Jodie Foster's. I saw this movie was made right between her 2 Oscar award winning performances, so my expectations were fairly high. ~~Unfortunately~~ **Unf0rtunately**, I thought the movie was ~~terrible~~ **terrib1e** and I'm still left wondering how she was ever persuaded to make this movie. The script is really ~~weak~~ **wea k**.

- After editing words (red), prediction changes from **100% of Negative** to **89% of Positive**.

Li et al (DNSS 2020). TextBugger: Generating Adversarial Text Against Real-world Applications

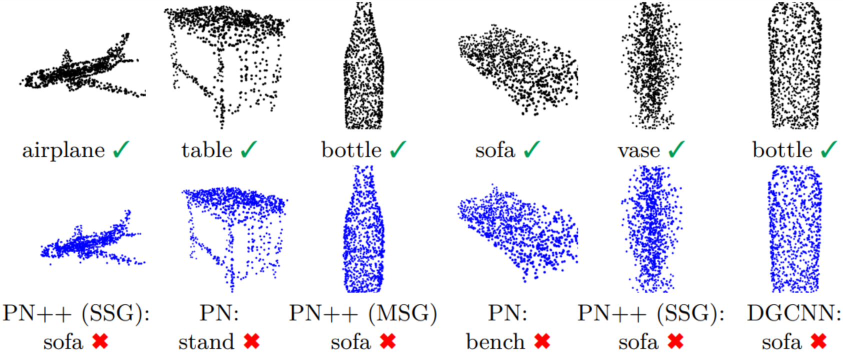
Adversarial attack on BERT-based sentiment classifier:

	Movie Review (Positive (POS) $\leftrightarrow$ Negative (NEG))
Original (Label: NEG)	The characters, cast in impossibly ***contrived situations***, are ***totally*** estranged from reality.
Attack (Label: POS)	The characters, cast in impossibly ***engineered circumstances***, are ***fully*** estranged from reality.
Original (Label: POS)	It cuts to the ***knot*** of what it actually means to face your ***scares***, and to ride the ***overwhelming*** **metaphorical wave** that life wherever it takes you.
Attack (Label: NEG)	It cuts to the ***core*** of what it actually means to face your ***fears***, and to ride the ***big*** **metaphorical wave** that life wherever it takes you.

- Changing a few words completely fools the BERT model

Ji et al (AAAI 2020). Is BERT really robust? a strong baseline for natural language attack on text classification and entailment

More Attacks: 3D Point Cloud Models

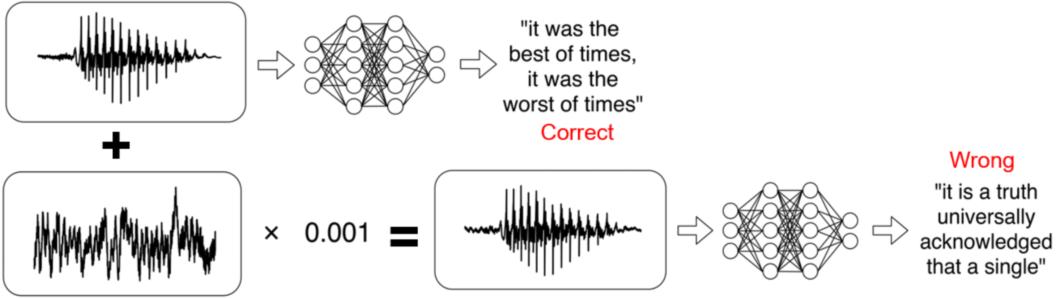


Adversarial attacks on multiple **3D Point Cloud** models by slightly perturbing the locations of the points

---

Hamdi et al (ECCV 2020). AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds

More Attacks: Audio Recognition Systems



Imperceptible adversarial examples can be generated to fool **Audio Recognition Systems** including Google Speech, Bing Speech, IBM Speech APIs, etc

---

Hadi et al (DNSS 2020). Practical Hidden Voice Attacks against Speech and Speaker Recognition Systems

Existing Tools for Adversarial Attacks

- Adversarial Robustness Toolbox (ART):
  `https://github.com/Trusted-AI/adversarial-robustness-toolbox`
- Foolbox Native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX
  `https://github.com/bethgelab/foolbox`
- CleverHans: `https://github.com/tensorflow/cleverhans`
- Advbox Family: `https://github.com/advboxes/AdvBox`
- ...

Existing Surveys for Adversarial Attacks

- Huang, Xiaowei, et al. "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability." Computer Science Review 37 (2020): 100270.
- Hao-Chen, Han Xu Yao Ma, et al. "Adversarial attacks and defenses in images, graphs and text: A review." International Journal of Automation and Computing 17.2 (2020): 151-178.
- Akhtar, Naveed, and Ajmal Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey." IEEE Access 6 (2018): 14410-14430.
- ...

What's the next?

- Adversarial attacks are important
  - **Understanding** the limitation, or potential safety risks of deep learning models
  - Providing a way to **practically evaluate** robustness performance of deep learning models under adversarial environments
  - Issues:
    - It cannot provide robustness guarantees in terms of excluding adversarial examples
    - Attacks alone cannot directly improve the robustness

- How can we improve its robustness? $\rightarrow$ **Part-2: Rectification**

- How to assess the adversarial robustness with provable guarantees $\rightarrow$ **Part-3: Verification**

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Rectification through Adversarial Training

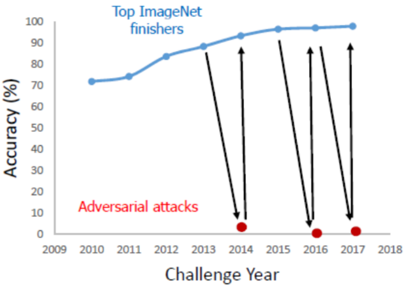
Rectification via Adversarial Defence

A fast growing research area:

- **Input denoising**
  ◇ e.g., Guo et al (2017); Buckman et al (2018); Liao et al (2018); Samangouei et al (2018); Bai et al (2019); etc.
- **Randomised smoothing**
  ◇ e.g., Lecuyer et al (2019); Li et al (2019); Cohen et al (2019); Salman et al (2019); Levine & Feizi (2020); Lee et al (2019); Teng et al (2020); Zhang et al (2020); etc.
- **Adversarial training**
  - **Training dataset augmentation**
    ◇ e.g., Goodfellow et al (2014); Shaham et al (2018); Sabour et al (2015); Kurakin et al (2016); Papernot et al (2016); Dezfooli et al (2016); etc.
  - **Robust optimisation**
    ◇ e.g., Goodfellow et al (2015); Madry et al (2017); Zhang et al (2019); Miyato et al (2018); Wang et al (2019); etc.

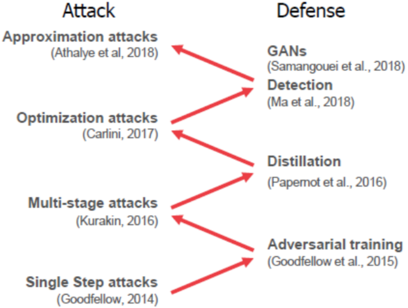
Attack vs. Defence: An Endless Game

Adversarial attacks cause a catastrophic reduction in ML capability



ImageNet Classification

Many defenses have been tried and failed to generalize to new attacks



Attack / Defense Cycle

@ DARPA's GARD programme

Adversarial Training Survives

- Athalye et al (ICML 2018). Obfuscated Gradients Give a False Sense of Security.
- Successful attack of 7 out of 9 defense in ICLR 2018
- The only survival is **adversarial training**

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%**~~ 0% Ilyas et al 2019
Madry et al. (2018)	CIFAR	0.031 (ℓ_∞)	47%
Na et al. (2018)	CIFAR	0.015 (ℓ_∞)	15%

Table of Contents

Introduction

Falsification through Adversarial Attack

- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training

- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification

- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Rectification through Adversarial Training

Adversarial Training

Adversarial Training via Robust Optimisation

- Madry et al (ICLR 2018). Towards Deep Learning Models Resistant to Adversarial Attacks.
- Idea: solving a minimax optimisation problem through SGD training

$$\min_{\theta}\{\mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{x'\in S_x} L(x',y;\theta)]\},$$

  - (x, y) - clean training data samples $x \in \mathbb{R}^n$ with labels $y \in [k]$ drawn from the dataset $\mathcal{D}$
  - $L(\cdot)$ - loss function with model parameter $\theta \in \mathbb{R}^m$
  - $x' \in \mathbb{R}^n$ - perturbed samples in a feasible region $S_x \triangleq \{z : z \in B(x,\epsilon) \cap [-1.0, 1.0]^n\}$
  - e.g., $B(z,\epsilon) \triangleq \{z : \|x - z\|_p \leq \epsilon\}$ - the ℓ_p-ball at center x with radius ϵ.

Adversarial Attacks as Inner Maximisation

- Outer minimisation can be simulated by SGD training

$$\min_{\theta}\{\frac{1}{N}\sum_{i=1}^{N}[\max_{x'\in S_x} L(x',y;\theta)]\},$$

- How to compute gradient of a maximisation?
  - Danskin's Theorem

$$\nabla_\theta \max L(x',y;\theta) = \nabla_\theta L(x^*,y;\theta)$$

  where $x^* = \arg\max L(x',y;\theta)$
- Inner maximisation $\max_{x'\in S_x} L(x',y;\theta)$ can be simulated by finding the worst-case adversarial attacks:
  - Fast Gradient Method (FGM)
  - Projected Gradient Method (PGM):

Fast Gradient Method (FGM)

- Goodfellow et al (2014). Explaining and harnessing adversarial examples. arXiv:1412.6572.
- Idea: Projecting perturbation onto the direction of gradient ascent of loss function
- Adversarial examples:

$$x^* = \arg\max_{x' \in S_x} \langle x' - x, \nabla_\theta L(x, y; \theta) \rangle$$

- For ℓ_∞-norm, FGM recovers the fast gradient sign method (FGSM) where each data point (x, y) is perturbed by the ϵ-normalised sign vector of the loss's gradient

$$x^* = x + \epsilon \cdot \mathrm{sgn}(\nabla_\theta L(x, y; \theta))$$

Projected Gradient Method (PGM)

- Kurakin et al (2016). Adversarial machine learning at scale. arXiv:1611.01236.
- Idea: Iterative gradient ascent to generate strongest adversarial examples, followed by projection back to the feasible region
- Updating rule:

$$x^{t+1} = \Pi_{S_x}(x^t + \alpha \cdot \mathrm{sgn}(\nabla_x L(x^t, y; \theta))),$$

where $\Pi_{S_x}(\cdot)$ projects the inputs onto the region S_x.

AT Variants

- FreeAT:
  - Shafahi et al (NeurIPS 2019). Adversarial training for free!
  - `https://github.com/ashafahi/free_adv_train/`
- YOPO:
  - Zhang et al (NeurIPS 2019). You only propagate once: Accelerating adversarial training via maximal principle.
  - `https://github.com/a1600012888/YOPO-You-Only-Propagate-Once`
- FreeLB:
  - Zhu et al (ICLR 2020). FreeLB: enhanced adversarial training for language understanding.
  - `https://github.com/zhuchen03/FreeLB`
- FastAT
  - Wong et al (ICLR 2020). Fast is better than free: Revisiting adversarial training.
  - `https://github.com/locuslab/fast_adversarial`

Pros and Cons of Adversarial Training

- Pros:
  - Empirical robustness (although no robust certificate)
  - Without affecting inference time (although increasing the training time)
  - Integratable to different threat models

- Cons:
  - Sacrifice accuracy to robustness
  - Dedicated to supervised learning
  - Rely very much on identifying local adversarial examples for specific threat models
  - No guarantees of generalisation performance

The min-max optimisation problem

$$\min_{\theta}\{\mathbb{E}_{(x,y)\sim\mathcal{D}}[\max_{x'\in S_x} L(x',y;\theta)]\},$$

where robustness is the goal.

How shall we deal with the following?

- Robustness vs Accuracy
- Supervised vs Semi-supervised Learning
- Local vs Global Information
- Robustness vs Generalisation

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Rectification through Adversarial Training

Distributional Robustness

- Sinha et al (ICLR 2018). Certifying some distributional robustness with principled adversarial training.
- Idea: Considering a Lagrangian penalty formulation of perturbing the underlying data distribution in a Wasserstein ball

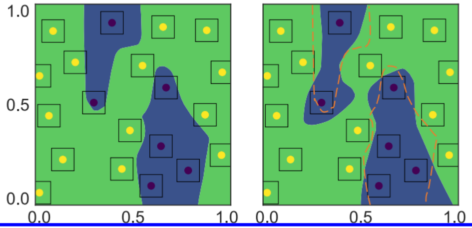
$$\begin{aligned} \min_{\theta} \sup_{P \in \mathcal{P}} \quad & \mathbb{E}_{(x,y)\sim P}[L(x,y;\theta) - \gamma W_c(P, P_0)] \\ s.t. \quad & W_c(P, P_0) \triangleq \inf_{M \in \Pi(P,P_0)} \mathbb{E}_M c(Z, Z') \end{aligned}$$

where P_0 is data-generating distribution, P is the perturbed distribution from P_0 such that $\mathcal{P} = \{P : W_c(P, P_0) \leq \rho\}$, $W_c(\cdot,\cdot)$ is Wasserstein metric, $c(Z, Z')$ is the transport cost from Z to Z', and M is certain measure.

- Zhang et al (ICML 2019). Theoretically principled trade-off between robustness and accuracy.
- Idea: optimizing a regularised surrogate loss

$$\min_{\theta}\{\mathbb{E}_{(x,y)\sim\mathcal{D}}[L(x,y;\theta)+\beta \max_{x'\in B(x,\epsilon)} \mathrm{KL}(f(x)||f(x'))]\},$$

  - **empirical loss minimisation**: maximise the natural accuracy
  - **regularization term**: push the decision boundary away from the data, so as to improve adversarial robustness



- Miyato et al (TPAMI 2018). Virtual adversarial training: a regularisation method for supervised and semi-supervised learning.
- Idea: regularise on local distributional smoothness (LDS)
- Virtual adversarial loss with "virtual" label: The distance of the conditional label distributions around each input data point against local perturbation

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}_{(x,y)\sim\mathcal{D}_l} L(x,y;\theta) + \alpha \mathbb{E}_{x_* \sim \mathcal{D}_l \cup \mathcal{D}_{ul}} LDS(x_*;\theta) \\ \text{s.t.} \quad & LDS(x_*;\theta) \triangleq D(p(\cdot|x_*;\hat{\theta})||p(\cdot|x_* + r_{\text{vadv}};\theta)) \\ & r_{\text{vadv}} = \arg\max_{||r||_2 \leq \epsilon} D(p(\cdot|x_*;\hat{\theta})||p(\cdot|x_* + r)) \end{aligned}$$

- Virtual adversarial direction: A direction of the adversarial perturbation that alter the output distribution at most.

Local vs Global Information

- Zhang and Wang (NeurIPS 2019). Defense against adversarial attacks using feature scattering-based adversarial training.
  - Motivation: Vanilla adversarial training generates adversarial examples one by one separately, without considering inter-sample relationship
  - Idea: Generating adversarial examples by perturbing the local neighborhood structure in an unsupervised fashion using feature scattering, and then performing model training with the generated adversarial examples
  - Feature scattering: Maximizing the feature matching distance between the clean samples and the perturbed samples in the latent space.
- Drawbacks
  - Feature scattering only considers the inter-sample relationship within the batch
  - Biased towards the decision boundary, which potentially corrupts the structure of the original data distribution

Question: How could global data manifold information play a role?

Exploiting Local & Global Information

- Robust optimization with f-divergence regularization

$$\begin{aligned} \min_{\theta} \quad & \{\mathbb{E}_{f_\theta(x^{adv}) \sim P_\theta^*}[l(f_\theta(x^{adv}), y)] + D_f(P_\theta^* || Q_\theta)\} \\ \text{s.t.} \quad & P_\theta^* = \arg\max_{P_\theta \in \mathcal{P}} [D_f(P_\theta || Q_\theta)] \end{aligned}$$

where $D_f(\cdot)$ is the f-divergence measure of two distributions, Q_θ is the underlying distribution of the latent features of clean samples, and P_θ is the underlying distribution of the latent features of adversarial perturbations.

- Feasible region for the latent distribution

$$\mathcal{P} = \{P : f_\theta(x') \sim P \quad \text{subject to} \quad \forall x \sim Q_0, x' \in B(x, \epsilon)\}$$

is induced by the set of perturbed examples through $f_\theta(\cdot)$.

- Idea: (1) Leverage a discriminator network for estimating the f-divergence between two distributions; (2) Generate Latent Manifold Adversarial Examples (LMAEs) to 'deceive' the latent manifold rather than fool the classifier

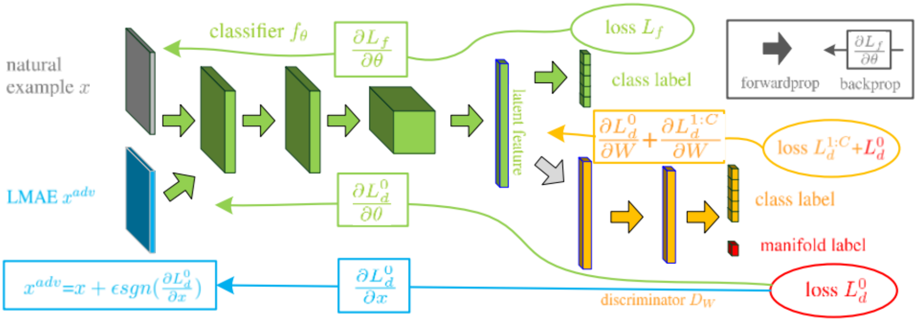
$$\min_{\theta}\Big\{\sum_{i=1}^{N}\underbrace{L(x_i^{adv},y_i;\theta)}_{L_f}+\sup_{W}\sum_{i=1}^{N}[\underbrace{\log D_W^0(f_\theta(x_i^{adv}))+(1-\log D_W^0(f_\theta(x_i)))}_{L_d^0}]$$

$$+\min_{W}[\underbrace{l(D_W^{1:C}(f_\theta(x_i)),y_i)+l(D_W^{1:C}(f_\theta(x_i^{adv})),y_i)}_{L_d^{1:C}}]\Big\}$$

$$\text{s.t.}\quad x_i^{adv}=\arg\max_{x_i'\in B(x_i,\epsilon)}[\log D_W^0(f_\theta(x_i'))+(1-\log D_W^0(f_\theta(x_i))].$$

where D_W denotes the discriminator network with parameter W, D_W^0 and $D_W^{1:C}$ are the different dimensions of the output of the discriminator

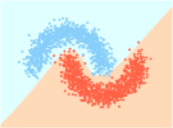
ATLD as an Adversarial Game

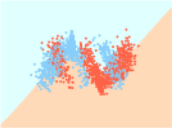


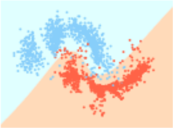
Adversarial game between a discriminator and a classifier:

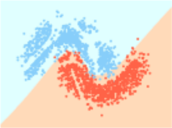
- Discriminator is learned to differentiate globally the latent distributions of the natural data and the perturbed counterpart
- Classifier is trained to recognize accurately the perturbed examples as well as enforcing the invariance between the two latent distributions

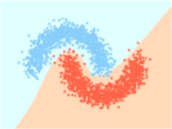
ATLD: Data Manifold and Decision Boundary


(a) Original Data


(b) PGD

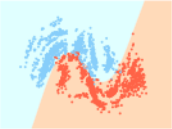

(c) Feature Scattering


(d) Ours


(e) Original


(f) PGD-AT


(g) Feature Scattering


(h) Ours

Qian et al (2021). Improving model robustness with latent distribution locally and globally. arXiv:2107.04401.

Introduction

Falsification through Adversarial Attack

Algorithms for Adversarial Attacks

More Examples of Adversarial Attacks

Rectification through Adversarial Training

Adversarial Training

Distributional Robustness

Robustness vs Generalisation

Robustness Verification

Over-approximation Techniques

Constraint Solving Techniques

Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Rectification through Adversarial Training

Robustness vs Generalisation

Standard regularisation techniques work for adversarial training to enhance generalisation performance

- Dropout
- Weight decay
- Data augmentation
- Early stopping

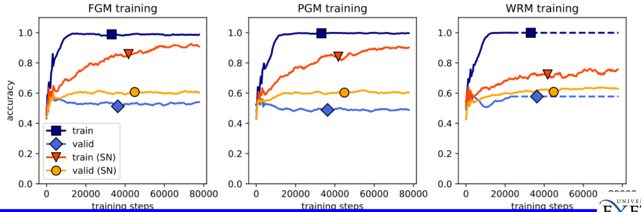
Q: Are there any weight regularisation techniques for generalisation that could be particularly suitable for adversarial training?

- Spectral normalisation
- Lipschitz regularisation
- Weight correction regularisation

- Miyato et al (ICLR 2018). Spectral normalization for generative adversarial networks.
  - Spectral normalisation: Normalising weight matrix by spectral norm, i.e., $\mathbf{W}_{\mathsf{SN}} = \frac{\mathbf{W}}{\sigma(\mathbf{W})}$ where

$$\sigma(\mathbf{W}) \triangleq \max_{\|\mathbf{x}\|_2 \leq 1} \|\mathbf{W}\mathbf{x}\|_2$$

- Farnia et al (ICLR 2019). Generalizable adversarial training via spectral normalization.



Lipschitz constraints under ℓ_2-norm are useful for provable adversarial robustness bounds, stable training, and Wasserstein distance estimation.

- Cisse et al (ICML 2017). Parseval networks: Improving robustness to adversarial examples.
  - Idea: to maintain weight matrices of linear and convolutional layers to be (approximately) Parseval tight frames (extensions of orthogonal matrices to non-square matrices).
- Li et al (NeurIPS 2019). Preventing gradient attenuation in lipschitz constrained convolutional networks.
  - Idea: to introduce convolutional gradient norm preserving networks with an efficient parameterisation of orthogonal convolutions to avoid the issues of loose bounds on the Lipschitz constant and computational intractability

- **Weight Correlation**: Given weight matrix $\mathbf{W}_l \in \mathbb{R}^{N_{l-1} \times N_l}$ of the l-th layer, the average weight correlation is defined as

$$\rho(\mathbf{W}_l) = \frac{1}{N_l(N_l-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{N_l} \frac{|\mathbf{w}_{l,i}^T \mathbf{w}_{l,j}|}{||\mathbf{w}_{l,i}||_2 ||\mathbf{w}_{l,j}||_2},$$

  where $\mathbf{w}_{l,i}$ and $\mathbf{w}_{l,j}$ are i-th and j-th column of $\mathbf{W}_l$, corresponding to the i-th and j-th neuron at l-th layer, respectively. Intuitively, $\rho(\mathbf{W}_l)$ is the average cosine similarity between weight vectors of any two neurons at the l-th layer.
- **Weight Correlation Regularisation**
  - Idea: Regularisation to constrain the average weight correlation between any two neurons so as to enhance generalisation performance

---

Jin et al (NeurIPS 2020). How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks.

- **Adversarial training** vs **norm regularisation**
  - Roth et al (NeurIPS 2020). Adversarial training is a form of data-dependent operator norm regularization.
  - **Insight**: ℓ_p-norm constrained projected gradient ascent based adversarial training with an ℓ_q-norm loss on the logits of clean and perturbed inputs is equivalent to data-dependent (p, q) operator norm regularization
- **Distributionally robust optimisation (DRO)** vs **regularisation**
  - Husain (NeurIPS 2020). Distributional robustness with IPMs and links to regularization and GANs.
  - **Insight**: DRO under any choice of Integral Probability Metrics (IPM) corresponds to a family of regularization penalties, which recover and improve upon existing results in the setting of Maximum Mean Discrepancy (MMD) and Wasserstein distances.

- **Generalisation error**:

$$\mathrm{GE} \triangleq |l(f_\theta(S), Y) - \hat{l}(f_\theta(S_d), Y_d)|$$

where $l(f_\theta(S), Y) \triangleq \mathbb{E}_{(x,y)\sim(S,Y)}[l(f_\theta(x), y)]$ and $\hat{l}(f_\theta(S_d), Y_d) \triangleq \frac{1}{|S_d|}\sum_{(x_d,y_d)\in S_d} l(f_\theta(x_d), y_d)$ with S_d, Y_d being the training data and the corresponding labels, respectively, and S, Y being the underlying data and label distributions, respectively.

- **Robust generalisation error**:

$$\mathrm{RGE} \triangleq |l(f_\theta(S^{adv}), Y) - \hat{l}(f_\theta(S_d^{adv}), Y_d)|$$

where S_d^{adv} and S^{adv} are the set of adversarial examples for the training set and its underlying distribution.

Robust Generalisation is Hard

- Schmidt et al (NeurIPS 2018). Adversarially robust generalization requires more data.
  - Sample complexity of robust learning $>>$ sample complexity of "standard" learning — the gap holds irrespective of training algorithms or models
- Yin et al (ICML 2019). Rademacher complexity for adversarially robust generalization.
  - Adversarial Rademacher complexity is larger than its natural counterpart
  - It has an unavoidable dimension dependence, unless the weight vector has bounded ℓ_1 norm
- Raghunathan et al (2019). Adversarial training can hurt generalization.
  - Adversarial training hurts generalisation even when the optimal predictor has both optimal standard and robust accuracy

Question: Why is robust generalisation hard to achieve and how to improve it?

Robust Generalisation Gap

Given the training set $S_d = \{x_i\}_{i=1}^n$ drawn from a distribution S with K classes, and the corresponding adversarial example set $S_d^{adv} = \{x_i^{adv}\}_{i=1}^n$ drawn from the underlying distribution S^{adv}, if the loss function $l(\cdot)$ of DNN f_θ is κ-Lipschitz, then for any $\delta > 0$, with the probability at least $1 - \delta$

$$\text{RGE} \leq \text{GE} + \frac{\kappa}{n} \sum_{i=1}^{K} \sum_{j \in N_i} \| d_\theta(x_j^{adv}) - \hat{d}_\theta(z, C_i) \|_2^2 + M \sqrt{\frac{2K \ln 2 + 2 \ln \frac{1}{\delta}}{n}}$$

where

$$d_\theta(x^{adv}) = f_\theta(x^{adv}) - f_\theta(x)$$

$$\hat{d}_\theta(z, C_i) = \mathbb{E}[f_\theta(z^{adv}) - f_\theta(z) | z \in C_i]$$

with N_i being the set of index of training data for class i, C_i the set of i^{th} class data of the whole set and z is data sampled from C_i with corresponding adversarial example z^{adv}, M the upper bound of loss of the whole data manifold S.

Adversarial Training with Shift Consistency Regularisation (AT-SCR)

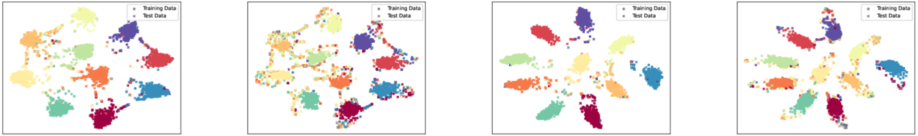
$$\min_{\theta} \quad \left\{ \sum_{i=1}^{n} [L(x_i^{adv}, y_i; \theta)] + \frac{\lambda}{n} \sum_{i=1}^{K} \sum_{j \in N_i} \widehat{\mathrm{SiC}}(x_j^{adv}, x_l, N_i) \right\},$$
$$\text{s.t.} \quad x_i^{adv} = \arg\max_{x_i' \in S_{x_i}} L(x_i', y_i; \theta).$$

where

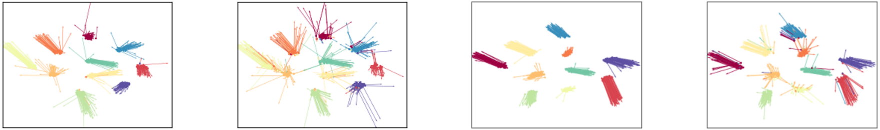
$$\widehat{\mathrm{SiC}}(x_j^{adv}, x_l, N_i) \triangleq \| d_\theta(x_j^{adv}) - \bar{d}_\theta(x_l, N_i) \|_2^2,$$

where $\bar{d}_\theta(x_l, N_i)$ is the average feature shifts over training data of class i, i.e.,

$$\bar{d}_\theta(x_l, N_i) = \frac{1}{|N_i|} \sum_{l \in N_i} (f_\theta(x_l^{adv}) - f_\theta(x_l)).$$



(a) FS w/ clean data (b) FS w/ adversarial data (c) FS-SCR w/ clean data (d) FS-SCR w/ adversarial data



(a) FS training shifts (b) FS test shifts (c) FS-SCR training shifts (d) FS-SCR test shifts

Zhang et al (ICML 2021). Towards Better Robust Generalization with Shift Consistency Regularization.

Concluding Remarks

- Distributional robustness is more preferable for adversarial training
  - trade-off between robustness and accuracy
  - both supervised and semi-supervised learning
  - both local and global information
- Robustness vs generalisation
  - Regularisation techniques could benefit both robustness and generalisation simultaneously
  - Robust generalisation requires rethinking latent dispersion of clean and adversarial examples

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

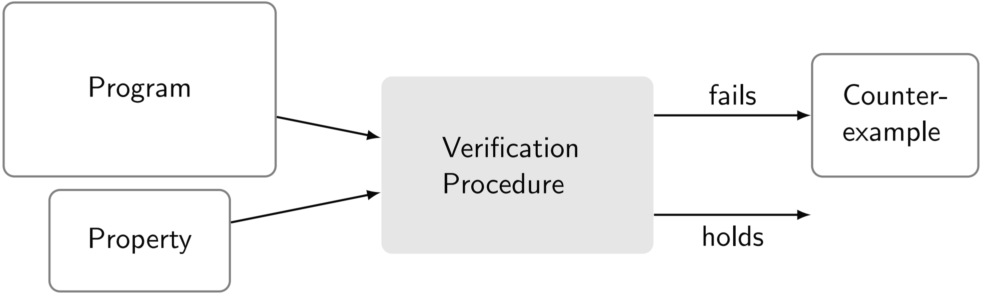
Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

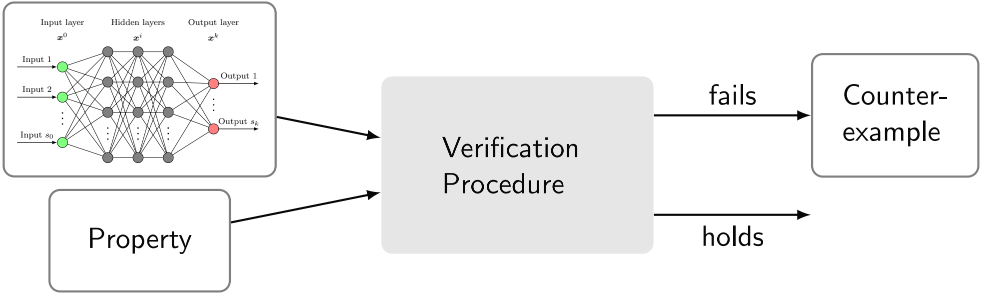
Verification in Practice

Conclusions and Future Directions

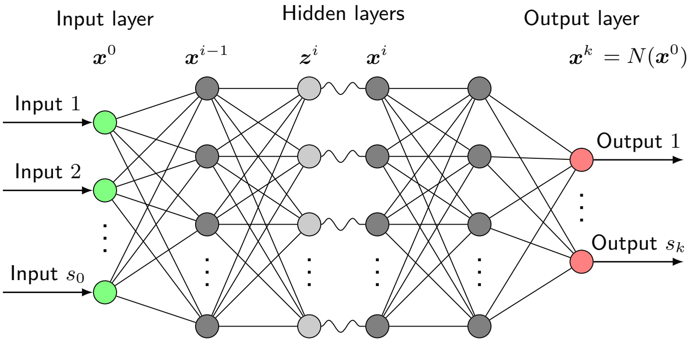
- formal guarantees



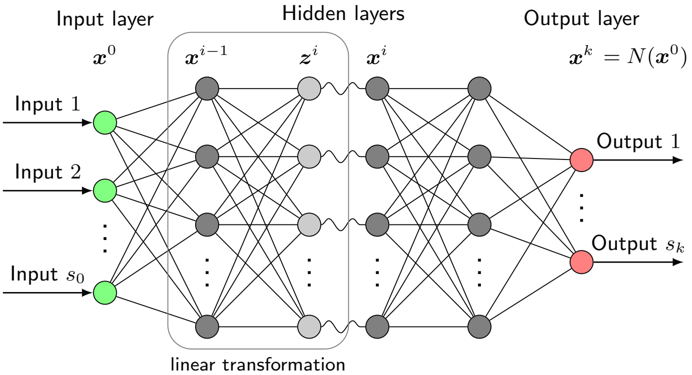
- formal guarantees



Neural Network Notation

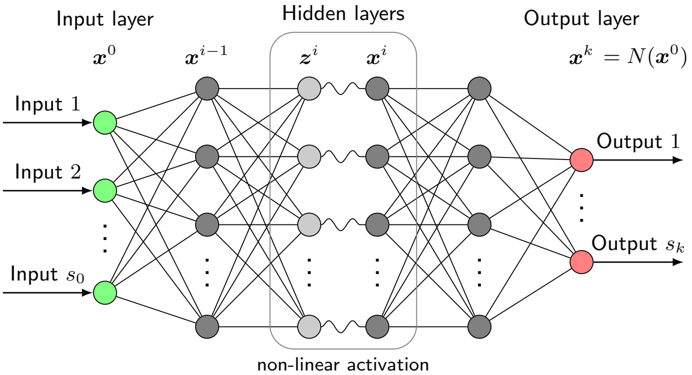


Neural Network Notation



- $\boldsymbol{z}^i$ are **preactivations**, $\boldsymbol{z}^i = W^i \boldsymbol{x}^{i-1} + b^i$

Neural Network Notation



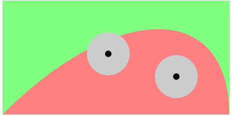
- $\boldsymbol{z}^i$ are **preactivations**, $\boldsymbol{z}^i = W^i \boldsymbol{x}^{i-1} + b^i$
- $\boldsymbol{x}^i$ are **postactivations**, $\boldsymbol{x}^i = \text{ReLU}(\boldsymbol{z}^i) = \max(0, \boldsymbol{z}^i)$

Properties to Verify

Generic input-output relation

$$\forall \boldsymbol{x}^0 \in \mathcal{I} \qquad N(\boldsymbol{x}^0) \in \mathcal{O}$$

- local robustness



- reachability
- safety
- semantic perturbations
  - rotation, translation, brightness and contrast, etc.

Given a network N, an input $\hat{\boldsymbol{x}} \in \mathbb{R}^{s_0}$, a perturbation radius r and a distance metric $||\cdot||_p$, decide whether

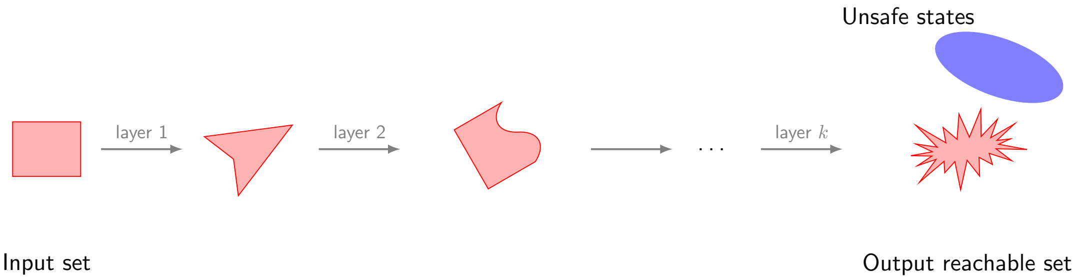
$$\text{arg}\max_i N(\boldsymbol{x}^0)_i = \text{arg}\max_i N(\hat{\boldsymbol{x}})_i$$

for all $\boldsymbol{x}^0$ such that $||\boldsymbol{x}^0 - \hat{\boldsymbol{x}}||_p \leq r$.

Here we focus on the infinity norm $||\boldsymbol{x}||_\infty := \max_i |\boldsymbol{x}_i|$.
$\Rightarrow \boldsymbol{x}^0 \in [\boldsymbol{l}^0, \boldsymbol{u}^0]$, where $\boldsymbol{l}_i^0 = \hat{\boldsymbol{x}}_i - r$ and $\boldsymbol{u}_i^0 = \hat{\boldsymbol{x}}_i + r$.

Verification is a difficult problem

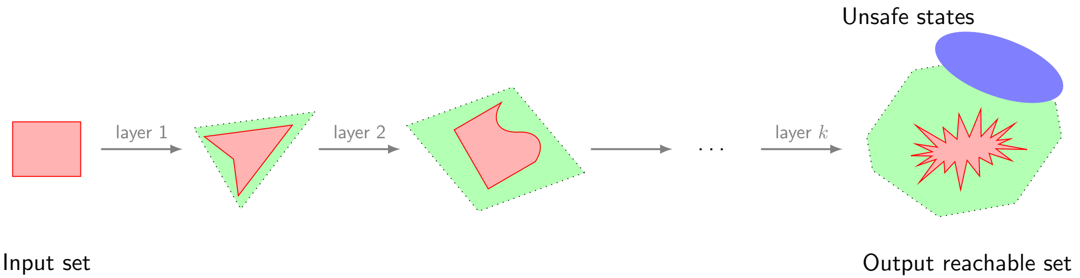
- exact reachability is NP-complete for ReLU networks



Challenges in Verification of Neural Networks

Verification is a difficult problem

- exact reachability is NP-complete for ReLU networks



- approximate methods offer better scalability

State-of-the-art Verification Approaches

- (Sound and) **incomplete** methods for general activations
  - Over-approximation
    1. Abstract Interpretation
    2. Estimation of output bounds
  - Global optimisation
- (Sound and) **complete** methods for piecewise-linear activations
  - Constraint solving approaches
    - SMT
    - MILP
  - Abstraction and iterative refinement

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

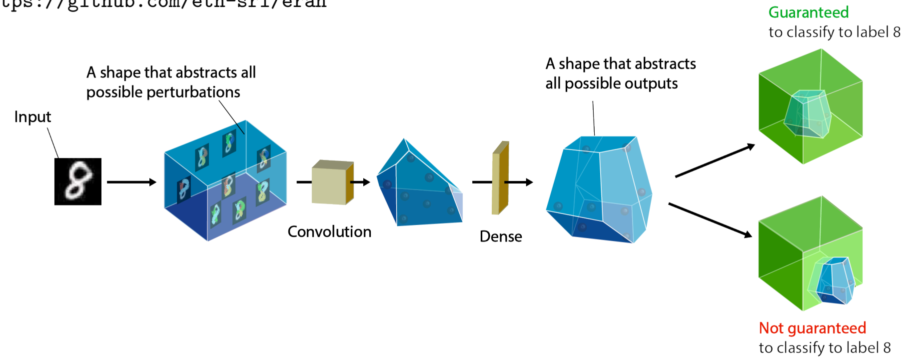
Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

https://github.com/eth-sri/eran



Gehr et al (S&P 2018). AI^2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation.
Singh et al (NeurIPS 2018). Fast and Effective Robustness Certification.
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.
Singh et al (ICLR 2019). Boosting Robustness Certification Of Neural Networks.

Bounds Computation

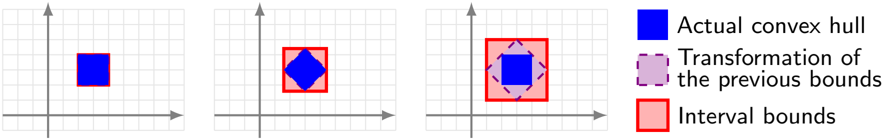
As the core of an over-approximation approach or as a pre-processing step.

Bounds Computation

As the core of an over-approximation approach or as a pre-processing step.

Interval propagation

- propagate intervals layer by layer



As the core of an over-approximation approach or as a pre-processing step.

Linear optimisation

- linearly relax non-linearities and solve the optimisation problems

$$\begin{array}{ll} \min \boldsymbol{x}_j^i & \qquad \max \boldsymbol{x}_j^i \\ \text{subject to } \dots & \qquad \text{subject to } \dots \end{array}$$

As the core of an over-approximation approach or as a pre-processing step.

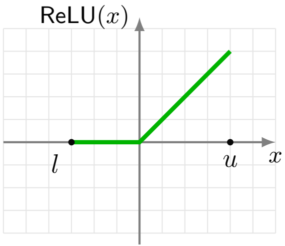
Symbolic interval propagation

- compute symbolic linear equations from the input variables

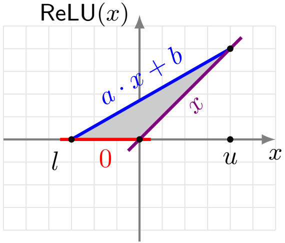
$$leq_i(\boldsymbol{x}^0) \leq \boldsymbol{x}^i \leq ueq_i(\boldsymbol{x}^0)$$

- concrete bounds are obtained by substituting the bounds for $\boldsymbol{x}^0$.

Linear Relaxation of Unstable ReLU



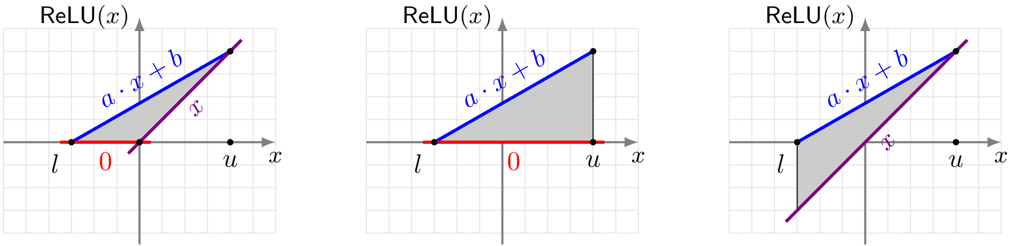
Linear Relaxation of Unstable ReLU



Upper bound equation $\Upsilon_{l,u,x} = a \cdot x + b$, where $a = \frac{u}{u-l}$ and $b = \frac{-lu}{u-l}$

Ehlers (ATVA 2017). Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks.

Linear Relaxation of Unstable ReLU

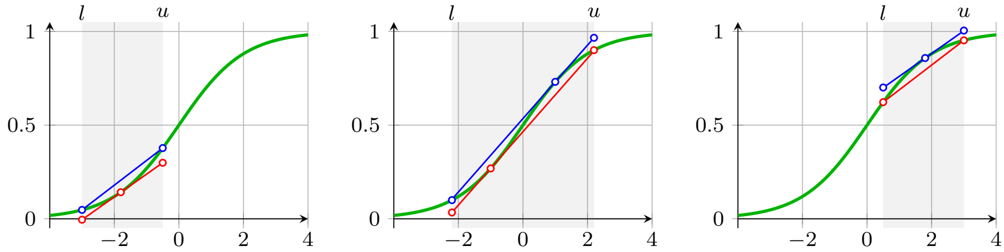


Upper bound equation $\Upsilon_{l,u,x} = a \cdot x + b$, where $a = \frac{u}{u-l}$ and $b = \frac{-lu}{u-l}$

Lower bound equation $\Lambda_{l,u,x} = \begin{cases} 0, & \text{if } u \leq -l \\ x, & \text{if } u > -l \end{cases}$

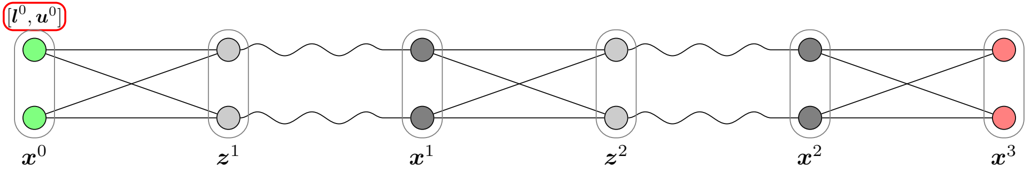
Ehlers (ATVA 2017). Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks.
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Linear Relaxation of S-shape Activation Functions



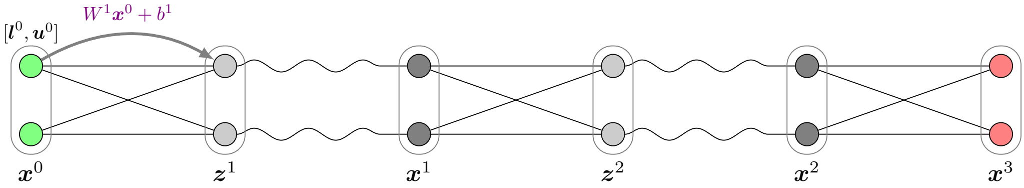
Zhang et al (Neurips 2018). Efficient Neural Network Robustness Certification with General Activation Functions.
Henriksen and Lomuscio (ECAI 2020). Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search.
Wu and Zhang (AAAI 2021). Tightening Robustness Verification of Convolutional Neural Networks with Fine-Grained Linear Approximation.

Symbolic Interval Propagation



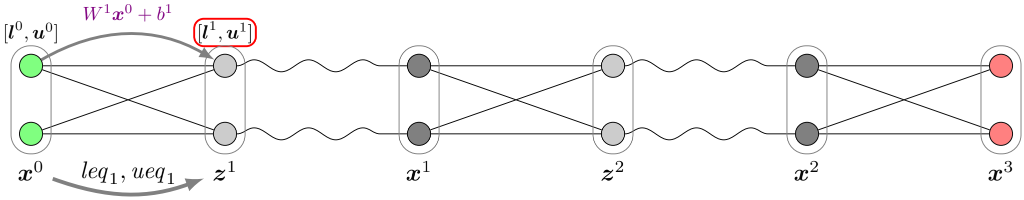
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation



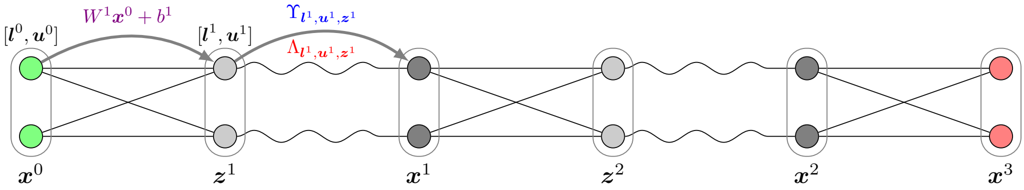
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation



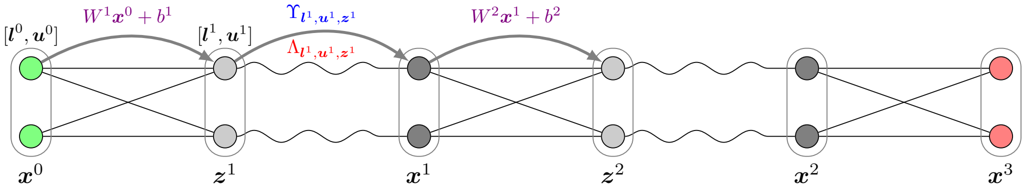
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation

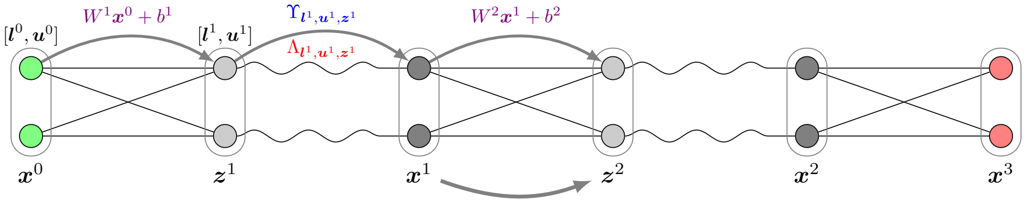


Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation

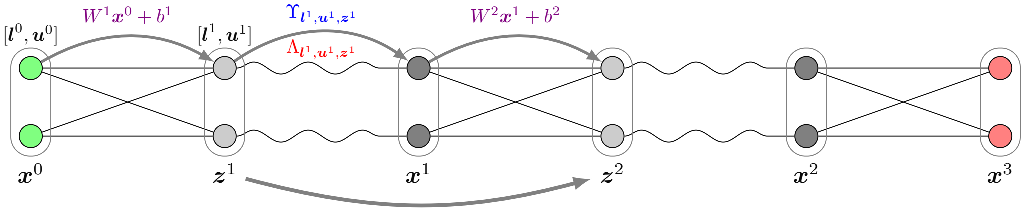


Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.



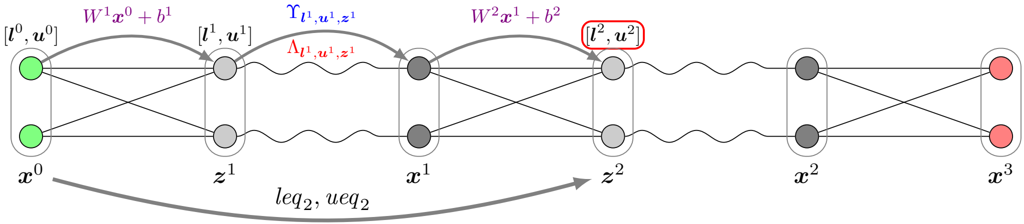
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation

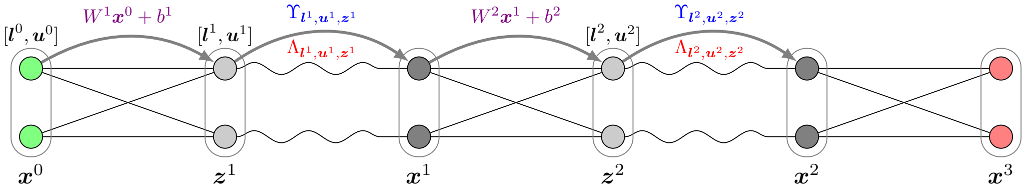


Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation

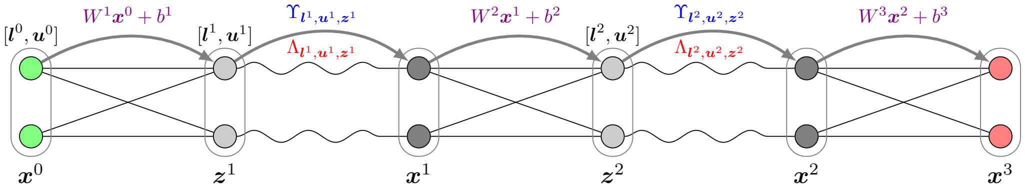


Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.



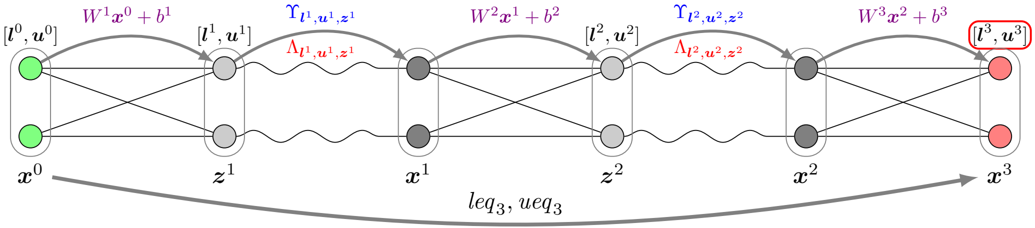
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation



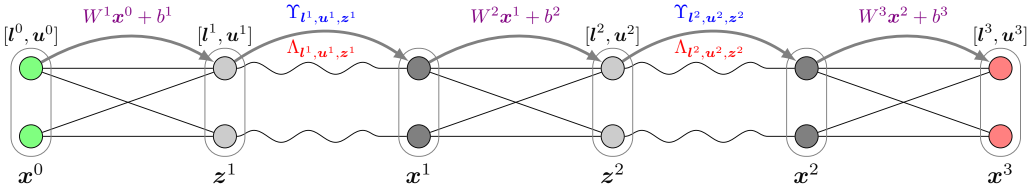
Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation



Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation

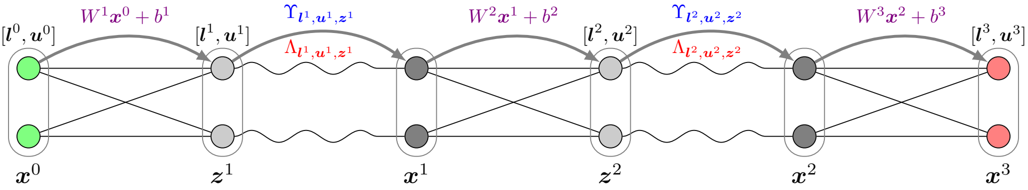


Substituting the equations **backwards** until the input layer

- current equation $Mx + o$, local equations $l(y)$ and $u(y)$ for x
- new lower bound equation $M^+ \cdot l(y) \; + \; M^- \cdot u(y) \; + \; o$
- new upper bound equation $M^+ \cdot u(y) \; + \; M^- \cdot l(y) \; + \; o$

Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.

Symbolic Interval Propagation



Substituting the equations **backwards** until the input layer

- current equation $Mx + o$, local equations $l(y)$ and $u(y)$ for x
- new lower bound equation $M^+ \cdot l(y) \ + \ M^- \cdot u(y) \ + \ o$
- new upper bound equation $M^+ \cdot u(y) \ + \ M^- \cdot l(y) \ + \ o$

Singh et al (POPL 2019). An Abstract Domain for Certifying Neural Networks.
Wang et al (NeurIPS 2018). Efficient Formal Safety Analysis of Neural Networks.
Weng et al (ICML 2018). Towards Fast Computation of Certified Robustness for ReLU Networks.

Introduction

Falsification through Adversarial Attack

Algorithms for Adversarial Attacks

More Examples of Adversarial Attacks

Rectification through Adversarial Training

Adversarial Training

Distributional Robustness

Robustness vs Generalisation

Robustness Verification

Over-approximation Techniques

Constraint Solving Techniques

Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

1. Encode as a set of linear constraints
   - the network
   - the input property
   - the **negation** of the output property
2. Check feasibility of the given set of constraints
   - If **feasible** $\Rightarrow$ a **counter-example** can be extracted from the satisfying assignment
   - Otherwise, it has been formally shown that the property is **satisfied**

Work for neural networks with **piecewise linear** activation functions.

Satisfiability of Boolean formulas with **special predicates**:

- e.g., linear inequalities
  - Simplex is a standard decision procedure for conjunctions of linear atoms.

Verification of Neural Networks:

1. Linear network constraints $\Rightarrow$ linear inequalities
2. Non-linear ReLU $\Rightarrow$ special ReLU constraint ReLU$(\boldsymbol{z}_j^i, \boldsymbol{x}_j^i)$
3. The simplex calculus is **extended** to handle ReLU constraints: **Reluplex**
4. SMT-based techniques are used to find a satisfying assignment

Katz et al (CAV 2017). Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks.
Katz et al (CAV 2019). The Marabou Framework for Verification and Analysis of Deep Neural Networks.

Feasibility of a set of linear inequalities over real and **integer-valued** variables.

Verification of Neural Networks:

- Piecewise linear non-linearities can be **directly** encoded using **binary** and **integer** variables.
- Off-the-shelf MILP solvers can be used to check feasibility of the encoding.
  - modern MILP solvers are very powerful

Lomuscio and Maganti (Arxiv 2017). An approach to reachability analysis for feed-forward ReLU neural networks.
Cheng, Nührenberg and Ruess (CAV 2017). Maximum resilience of artificial neural networks.
Fischetti and Jo (Constraints 2018). Deep neural networks and mixed integer linear optimization.
Tjeng, Xiao and Tedrake (ICLR 2019). Evaluating Robustness Of Neural Networks With Mixed Integer Programming.

- **Weighted sum**

$$z^i = W^i x^{i-1} + b^i$$

- **ReLU constraint** $x^i_j = \text{ReLU}(z^i_j)$ when $l^i_j < 0 < u^i_j$

$$\begin{array}{lll} x^i_j \geq 0 & x^i_j \leq u^i_j \cdot \delta^i_j & \delta^i_j = 0 \Rightarrow x^i_j = 0, \text{ inactive} \\ x^i_j \geq z^i_j & x^i_j \leq z^i_j - l^i_j \cdot (1 - \delta^i_j) & \delta^i_j = 1 \Rightarrow x^i_j = z^i_j, \text{ active} \end{array}$$

- **Input property** $||x^0 - \hat{x}||_\infty \leq r$

$$\hat{x}_j - r \leq x^0_j \leq \hat{x}_j + r$$

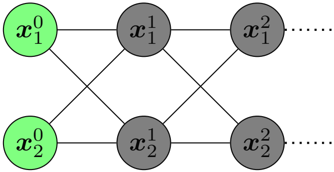
- (**Negation of**) **output property** $\arg\max_i x^k_i = \arg\max_i N(\hat{x})_i = c$

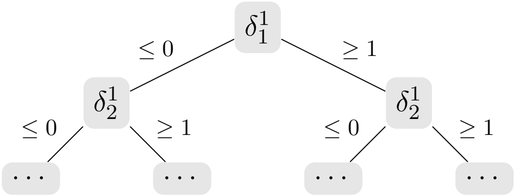
$$\bigvee_{j \neq c} x^k_j \geq x^k_c : \qquad \begin{array}{l} \beta_1 + \cdots + \beta_{s_k} = 1, \quad \beta_c = 0 \\ (\beta_j = 1) \Rightarrow x^k_j \geq x^k_c, \quad j \neq c \end{array}$$

Tjeng, Xiao and Tedrake (ICLR 2019). Evaluating Robustness Of Neural Networks With Mixed Integer Programming.

Branch-and-bound Procedure

1. Solve the linear relaxation of the program (integer variables can take real values).
2. If the solution satisfies all integer constraints $\rightarrow$ terminate.
3. Otherwise, branch on an integer variable with fractional value.
4. Repeat for each sub-problem.

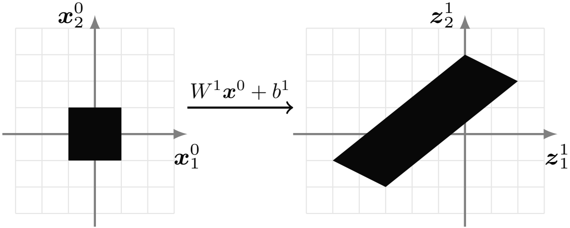




Unstable nodes n_q^i and n_r^j are in a **dependency relation**,
if fixing n_q^i in some stable state, fixes n_r^j in a stable state as well, and the other way around.

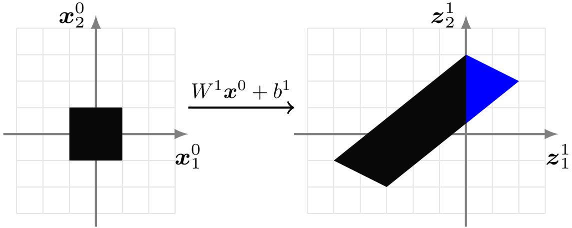
Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

Unstable nodes n_q^i and n_r^j are in a **dependency relation**,
if fixing n_q^i in some stable state, fixes n_r^j in a stable state as well, and the other way around.



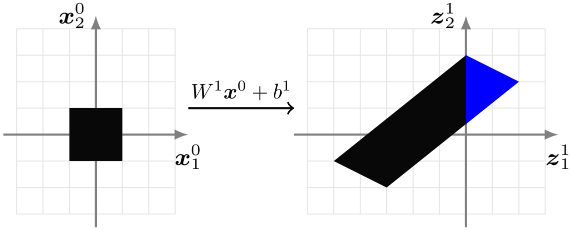
Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

Unstable nodes n_q^i and n_r^j are in a **dependency relation**,
if fixing n_q^i in some stable state, fixes n_r^j in a stable state as well, and the other way around.



Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

Unstable nodes n_q^i and n_r^j are in a **dependency relation**,
if fixing n_q^i in some stable state, fixes n_r^j in a stable state as well, and the other way around.

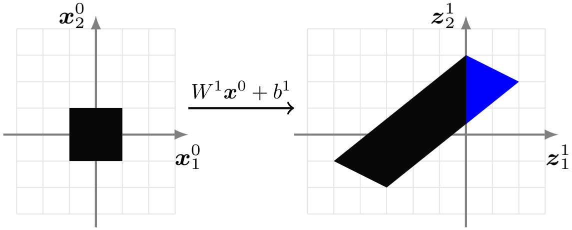


Dependency between n_1^1 and n_2^1: $(\delta_1^1 = 1) \rightarrow (\delta_2^1 = 1)$.

Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

Dependency Relation between ReLU Nodes

Unstable nodes n_q^i and n_r^j are in a **dependency relation**,
if fixing n_q^i in some stable state, fixes n_r^j in a stable state as well, and the other way around.

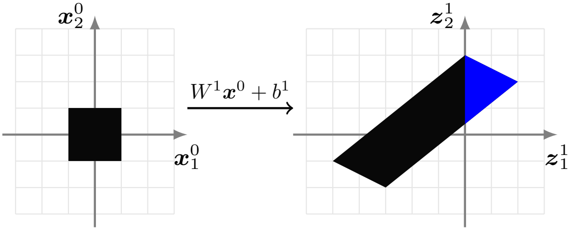


Dependency between n_1^1 and n_2^1: $(\delta_1^1 = 0) \vee (\delta_2^1 = 1)$. As MILP constraint: $\boxed{(1 - \delta_1^1) + \delta_2^1 \geq 1}$.

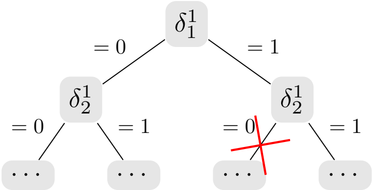
Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

Dependency Relation between ReLU Nodes

Unstable nodes n_q^i and n_r^j are in a **dependency relation**,
if fixing n_q^i in some stable state, fixes n_r^j in a stable state as well, and the other way around.



Reduction of the search space.

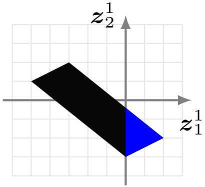


Dependency between n_1^1 and n_2^1: $(\delta_1^1 = 0) \vee (\delta_2^1 = 1)$. As MILP constraint: $\boxed{(1 - \delta_1^1) + \delta_2^1 \geq 1}$.

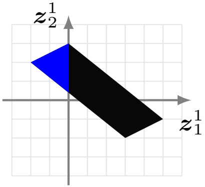
Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

Intra- and Inter-layer dependencies

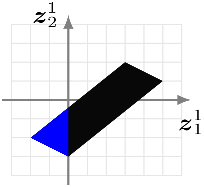
Intra-layer dependencies



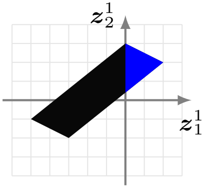
$(\delta_1^1 = 1) \to (\delta_2^1 = 0)$



$(\delta_1^1 = 0) \to (\delta_2^1 = 1)$

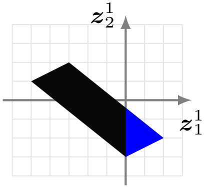


$(\delta_1^1 = 0) \to (\delta_2^1 = 0)$

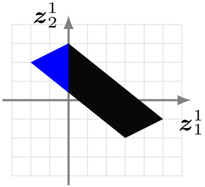


$(\delta_1^1 = 1) \to (\delta_2^1 = 1)$

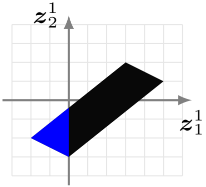
Intra-layer dependencies



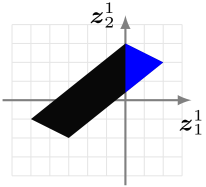
$(\delta_1^1 = 1) \to (\delta_2^1 = 0)$



$(\delta_1^1 = 0) \to (\delta_2^1 = 1)$

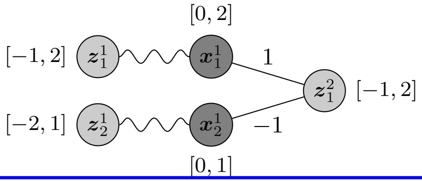


$(\delta_1^1 = 0) \to (\delta_2^1 = 0)$

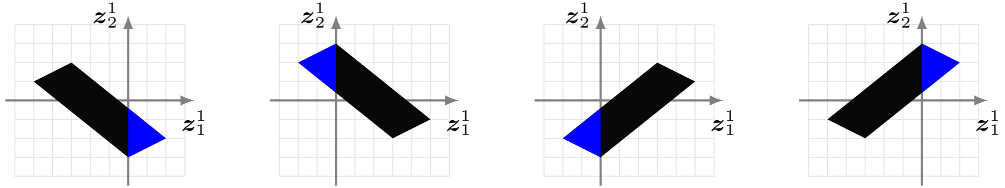


$(\delta_1^1 = 1) \to (\delta_2^1 = 1)$

Inter-layer dependencies

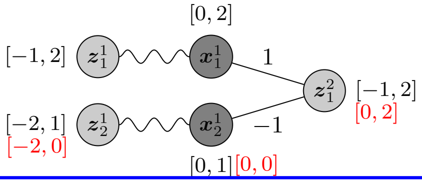


Intra-layer dependencies



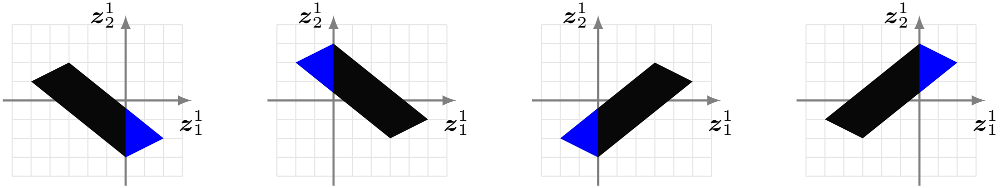
$(\delta_1^1 = 1) \rightarrow (\delta_2^1 = 0)$ $(\delta_1^1 = 0) \rightarrow (\delta_2^1 = 1)$ $(\delta_1^1 = 0) \rightarrow (\delta_2^1 = 0)$ $(\delta_1^1 = 1) \rightarrow (\delta_2^1 = 1)$

Inter-layer dependencies



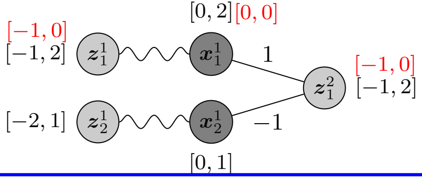
$$(\delta_2^1 = 0) \rightarrow (\delta_1^2 = 1)$$

Intra-layer dependencies



$(\delta_1^1 = 1) \to (\delta_2^1 = 0)$ $(\delta_1^1 = 0) \to (\delta_2^1 = 1)$ $(\delta_1^1 = 0) \to (\delta_2^1 = 0)$ $(\delta_1^1 = 1) \to (\delta_2^1 = 1)$

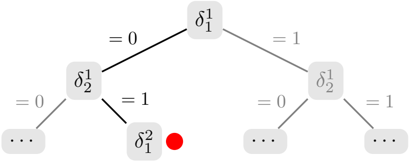
Inter-layer dependencies



$$(\delta_1^1 = 0) \to (\delta_1^2 = 0)$$

Inserting Dependency Cuts

1. Stop the branch-and-bound procedure at runtime.



2. Compute the dependencies given the partial assignment to δ_j^i.

$$\text{e.g., } (\delta_1^2 = 0) \vee (\delta_2^2 = 1) \text{ when } \delta_1^1 = 0 \text{ and } \delta_2^1 = 1$$

3. Add the dependencies as MILP constraints to the MILP formulation.

$$(1 - \delta_1^2) + \delta_2^2 + \underbrace{\delta_1^1 + (1 - \delta_2^1)}_{\text{0 under the current branch}} \geq 1$$

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

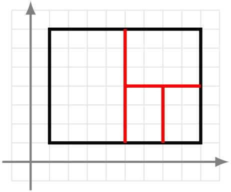
Conclusions and Future Directions

1. Verify using a fast incomplete method
2. If property holds $\rightarrow$ return **success**
3. If a counter-example found $\rightarrow$ return **failure**
4. If unknown $\rightarrow$ **refine** the verification problem
5. Repeat for each sub-problem

Refinement for over-approximation based abstraction:

- Input domain splitting
- ReLU node splitting

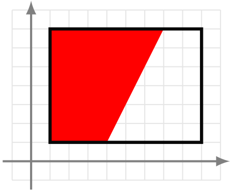
1. Bisect the input interval along one of the dimensions
   - smaller input intervals $\Rightarrow$ smaller over-approximation error.
2. Heuristics for choosing the dimension to split.
3. Works well for low dimensional inputs.
4. Can be used with arbitrary activation functions to produce better output bounds.
5. Can be used in conjunction with constraint solving approaches.



Wang et al (NeurIPS 2018). Formal Security Analysis of Neural Networks using Symbolic Intervals.
Rubies-Royo et al (Arxiv 2019). Fast Neural Network Verification via Shadow Prices.
Katz et al (CAV 2019). The Marabou Framework for Verification and Analysis of Deep Neural Networks.
Botoeva et al (AAAI 2020). Efficient Verification of ReLU-based Neural Networks via Dependency Analysis.

ReLU Node Splitting

1. Stabilise an unstable ReLU node
   - no need for linear relaxation $\Rightarrow$ smaller over-approximation error.
2. Heuristics for choosing the node to split.
3. Works well for high dimensional inputs.
4. Might require using an LP solver.



Wang et al (NeurIPS 2018). Efficient Formal Safety Analysis of Neural Networks.
Henriksen and Lomuscio (ECAI 2020). Efficient Neural Network Verification via Adaptive Refinement and Adversarial Search.
Bak (VNN 2020). Execution-Guided Overapproximation (EGO) for Improving Scalability of Neural Network Verification.
Henriksen and Lomuscio (IJCAI 2021). DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis.
Kouvaros and Lomuscio (IJCAI 2021). Towards Scalable Complete Verification of ReLU Neural Networks via Dependency-based Branching.

Other Abstraction Methods

- Ashok et al (ATVA 2020). DeepAbstract: Neural Network Abstraction for Accelerating Verification.
- Elboher, Gottschlich and Katz (CAV 2020). An Abstraction-Based Framework for Neural Network Verification.
- Prabhakar and Rahimi Afzal (NeurIPS 2019). Abstraction based Output Range Analysis for Neural Networks.
- Sotoudeh and Thakur (Arxiv 2020). Abstract Neural Networks.

Concluding Remarks

- Scalability remains the main concern
- Holistic approach to training and verification
  - models that are easier to verify
- Other kind of verification properties

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

Conclusions and Future Directions

Verification in Practice

What are the practical needs of verification?

- More properties to be verified.
  - robustness – local property for input perturbation
  - generalisation – global property for unseen data
  - security properties such as backdoor
- Different ways of expressing whether or not a model is dependable.
  - confirm whether or not a property holds
  - finding counterexamples
  - statistical evaluation
  - lower/upper bounds of a certain quantity

What are the practical needs of verification?

- Scalable to work with real-world neural networks
  - different types of layers and activation functions
  - large network (depth, width, etc)
- Concerns all inputs that may appear in operational time
  - Reliability, which describes the ability of a system or component to function under stated conditions for a specified period of time.

Approaches towards making verification practical

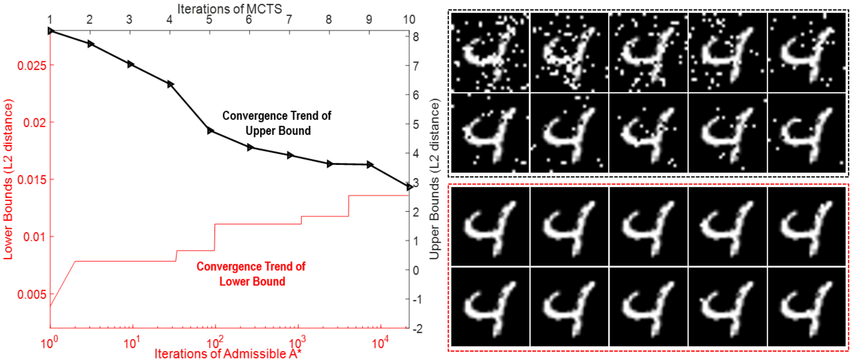
To be included in this tutorial:

- For local properties such as robustness
  - Global optimisation based methods – converging bounds
  - Sampling based methods – statistical bounds
  - Testing methods – use metrics to decide if the tests are sufficient
- For reliability
  - Assessment method based on the production of (global) generalisation and (local) robustness

Verification in Practice

$\hookrightarrow$Converging Bounds Methods

General idea of converging bounds method



[Huang et al., 2017] (CAV2017) Safety verification of deep neural networks.

[Wicker et al., 2018] (TACAS2018) Feature-guided black-box safety testing of deep neural networks

[Ruan et al., 2018] (IJCAI2018) Reachability Analysis of Deep Neural Networks with Provable Guarantees.

[Ruan et al., 2019] (IJCAI2019) Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance,

[Wu et al., 2020] (Theoretical Computer Science, 2020) A game-based approximate verification of deep neural networks with provable guarantees.

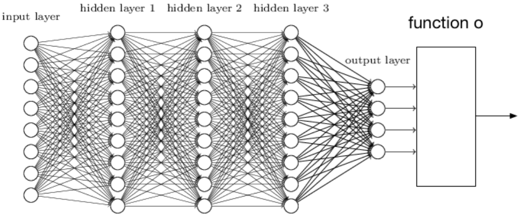
The following layers are Lipschitz continuous:

- convolutional with ReLU activation functions,
- fully connected layers with ReLU activation functions,
- max pooling
- contrast-normalization
- softmax (proved in this paper)
- sigmoid (proved in this paper)
- Hyperbolic tangent (proved in this paper)

Cover all layers used in e.g., image classification networks.

Let $o : [0,1]^m \rightarrow \mathbb{R}$ be a Lipschitz continuous function statistically evaluating the outputs of the network.

Connect the network f with function o, i.e., o(f(x))



Let $X' \subseteq [0,1]^n$ be an input subspace and $f : \mathbb{R}^n \to \mathbb{R}^m$ a network. The reachability of f over the function o under an error tolerance $\epsilon \geq 0$ is a set $R(o, X', \epsilon) = [l, u]$ such that

$$l \geq \inf_{x' \in X'} o(f(x')) - \epsilon \text{ and } u \leq \sup_{x' \in X'} o(f(x')) + \epsilon. \tag{4}$$

We write $u(o, X', \epsilon) = u$ and $l(o, X', \epsilon) = l$ for the upper and lower bound, respectively.

The instantiation of the o function will enable us to express several problems with a single formalism.

- output range analysis
- safety/robustness verification
- robustness comparison between networks and input subregions

- (Safety Definition) A network f is safe with respect to an input x and an input subspace $X' \subseteq [0,1]^n$ with $x \in X'$ if

$$\forall x' \in X' : \arg\max_j c_j(x') = \arg\max_j c_j(x) \tag{5}$$

- (Instantiation for safety) A network f is safe with respect to x and X' s.t. $x \in X'$ if and only if

$$u(\oplus, X', \epsilon) \leq 0$$

where $j = \arg\max_j c_j(x)$, $\oplus(c_1, ..., c_m) = \max_{i \in \{1..m\}}(\Pi_i(c_1, ..., c_m) - \Pi_j(c_1, ..., c_m))$. The error bound of the safety decision problem by this reduction is 2ϵ.

Let $w = o \cdot f$. The computation of the minimum value is reduced to solving the following optimization problem with guaranteed convergence to the global minimum.

$$\min_{x} \quad w(x), \quad s.t. \quad x \in [a,b]^n \tag{6}$$

The maximization problem can be transferred into a minimization problem.

Design another continuous function $h(x, y)$, which serves as a lower bound of the original function $w(x)$. Specifically, we need

$$h(x, y) \leq w(x), \ \forall x, y \in [a, b]^n, \ h(x, x) = w(x) \tag{7}$$

Furthermore, for $i \geq 0$, we let $\mathcal{Y}_i = \{y_0, y_1, ..., y_i\}$ be a finite set containing $i + 1$ points from the input space $[a, b]^n$, and let $\mathcal{Y}_i \subseteq \mathcal{Y}_k$ when $k > i$, then we can define a function $H(x; \mathcal{Y}_i) = \max_{y \in \mathcal{Y}_i} h(x, y)$ which satisfies the following relation:

$$H(x; \mathcal{Y}_i) < H(x; \mathcal{Y}_k) \leq w(x) \tag{8}$$

We use $l_i = \inf_{x \in [a,b]^n} H(x; \mathcal{Y}_i)$ to denote the minimum value of $H(x; \mathcal{Y}_i)$ for $x \in [a,b]^n$. Then we have

$$l_0 < l_1 < ... < l_{i-1} < l_i \leq \inf_{x \in [a,b]^n} w(x)$$

Similarly, we need a sequence of upper bounds u_i to have

$$l_0 < ... < l_i \leq \inf_{x \in [a,b]^n} w(x) \leq u_i < ... < u_0 \tag{9}$$

By Expression (9), we can have the following:

$$\lim_{i \to \infty} l_i = \min_{x \in [a,b]^n} w(x) \text{ and } \lim_{i \to \infty} (u_i - l_i) = 0 \tag{10}$$

Let $h(x, y) = w(y) - K|x - y|$

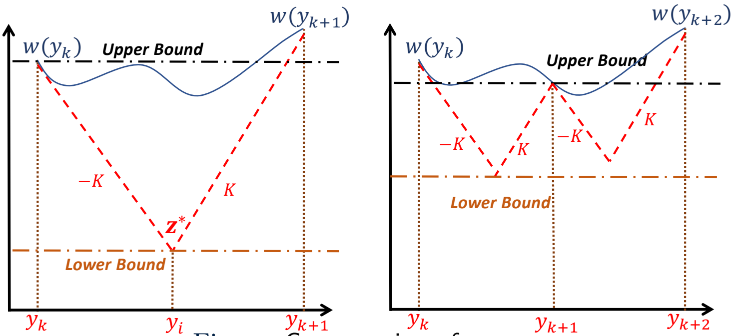


Figure: Computation of next y_i

NN ID	Layer No.	Neutron No.	Time by SHERLOCK	Time by Reluplex	Our method
N-0	1	100	1.9s	1m 55s	0.4s
N-1	1	200	2.4s	13m 58s	1.0s
N-2	1	500	17.8s	Timeout	6.8s
N-3	1	500	7.6s	Timeout	5.3s
N-4	1	1000	7m 57.8s	Timeout	1.8s
N-5	6	250	9m 48.4s	Timeout	15.1s

Figure: Comparison with SHERLOCK and Reluplex

Computational complexity is NP-complete over the input dimension, instead of number of neurons.

Verification in Practice

$\hookrightarrow$Sampling-based Methods

CLEVER [Weng et al., 2018]

- A lower bound of L_p minimum adversarial distortion β_L
- Extreme Value Theory ensures that the maximum value of random variables can only follow one of the three extreme value distributions.

[Weng et al., 2018] (ICLR2018) Evaluating The Robustness Of Neural Networks: An Extreme Value Theory Approach

- a naive Monte Carlo sampling does not work well for high-dimensional problems.
- an adaptation of multi-level splitting, a Monte Carlo approach for estimating the probability of rare events.

[Webb et al., 2019] (ICLR2019) A Statistical Approach To Assessing Neural Network Robustness

Verification in Practice

↪Software Testing Methods

- Well established in many industrial standard for software used in safety critical systems, such as ISO26262 for automotive systems and DO 178B/C for avionic systems.

- Coverage Metrics
  - structural coverage
  - scenario coverage
- Test Case Generation Methods
  - fuzzing
  - symbolic execution, etc

- to determine if the generated test cases include bugs.

- Industrial standards need to be upgraded

- A few Coverage Metrics [Pei et al., 2017, Sun et al., 2019]
- A few Test Case Generation Methods [Sun et al., 2018]

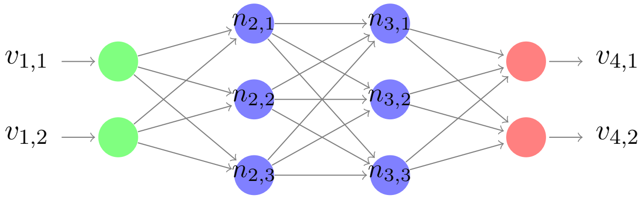
- Use a set of generated test cases to either finding bugs or evaluating the performance of a neural network

[Pei et al., 2017] (SOSP2017) DeepXplore: Automated Whitebox Testing of Deep Learning Systems.
[Sun et al., 2019] (EMSOFT2019) Structural Test Coverage Criteria for Deep Neural Networks.
[Sun et al., 2018] (ASE2018) Concolic Testing for Deep Neural Networks.

Coverage Metrics

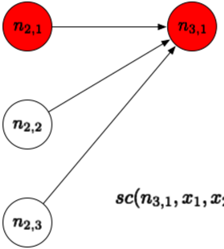
- Neuron Coverage [Pei et al., 2017]: to make sure that all neurons have been activated in at least one of the test cases
- boundary coverage : to make sure the boundary values of each neuron is reached.
- MC/DC coverage [Sun et al., 2019]: to make sure that every neuron in a layer can independently activate the neurons in the next layer.

The core idea of our criteria is to ensure that not only the presence of a feature needs to be tested but also the effects of less complex features on a more complex feature must be tested.



For example, check the impact of $n_{2,1}, n_{2,2}, n_{2,3}$ on $n_{3,1}$.

(Sign Change of a neuron) Given a neuron $n_{k,l}$ and two test cases x_1 and x_2, we say that the sign change of $n_{k,l}$ is exploited by x_1 and x_2, denoted as $sc(n_{k,l}, x_1, x_2)$, if $sign(v_{k,l}[x_1]) \neq sign(v_{k,l}[x_2])$.



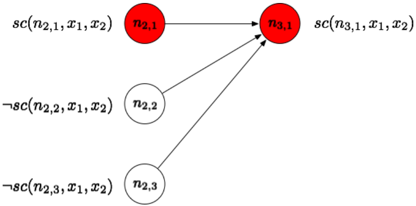
$$sc(n_{3,1}, x_1, x_2) \equiv sign(v_{3,1}[x_1]) \neq sign(v_{3,1}[x_2])$$

MC/DC for DNNs – Sign-Sign Cover, or SS Cover

A neuron pair $(n_{k,i}, n_{k+1,j})$ are two neurons in adjacent layers k and $k+1$ such that $1 \leq k \leq K-1$, $1 \leq i \leq s_k$, and $1 \leq j \leq s_{k+1}$.

A neuron pair $\alpha = (n_{k,i}, n_{k+1,j})$ is SS-covered by two test cases x_1, x_2, denoted as $SS(\alpha, x_1, x_2)$, if the following conditions are satisfied by the network instances $\mathcal{N}[x_1]$ and $\mathcal{N}[x_2]$:

- $sc(n_{k,i}, x_1, x_2)$;
- $\neg sc(n_{k,l}, x_1, x_2)$ for all $n_{k,l} \in P_k \setminus \{i\}$;
- $sc(n_{k+1,j}, x_1, x_2)$.



Then, what is the state-of-the-art on DNN Verification?

- Robustness

What is the actual need for certification?

- Reliability, e.g., the probability of no failure at all in the next prediction.

Verification in Practice

$\hookrightarrow$Reliability Assessment

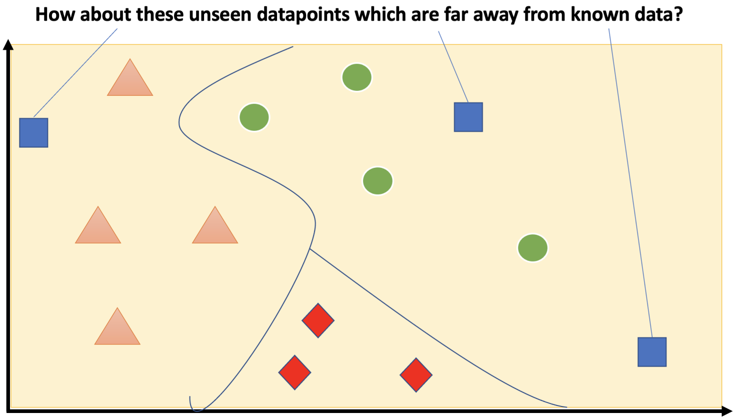
Reliability

- Reliability is required by industrial standards
  - consider e.g., the probability of no failure at all for the next 10^k input
  - safety integrity levels (SIL1 - SIL4), as in IEC 61508 standard for "Functional Safety of Electrical/Electronic/ Programmable Electronic Safety-related Systems"

- From Robustness to Reliability? [Zhao et al., 2020]
  - We do not know what the next input will be.
  - Generalisability!

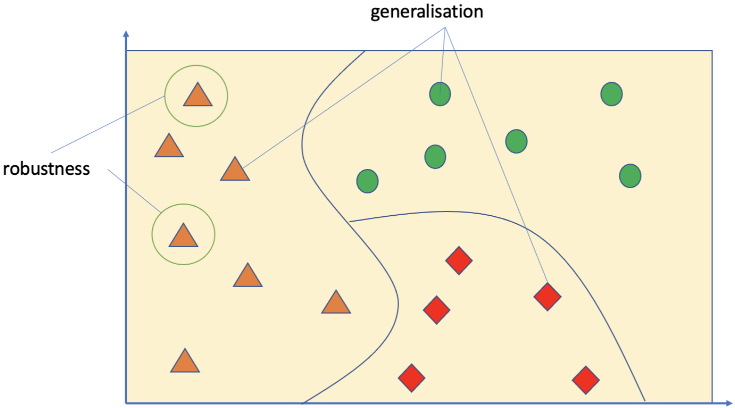
[Zhao et al., 2020] (SafeCOMP2020) A Safety Framework for Critical Systems Utilising Deep Neural Networks

What is generalisation failure?



Note: far away from known data $\neq$ far away from boundary

What do we already have?



What is reliability in the context of deep learning?

$$\text{Reliability} = \text{Generalisation} \times \text{Robustness}$$

i.e., we have the following definition for reliability [Zhao et al., 2021]

$$\lambda := \sum_{x \in \mathcal{D}} I_{\{x \text{ causes a failure}\}}(x) Op(x) \tag{11}$$

where $Op(x)$ captures the uncertainty of "which one would be the next input".

[Zhao et al., 2021] (AISafety2021) Assessing the Reliability of Deep Learning Classifiers Through Robustness Evaluation and Operational Profiles

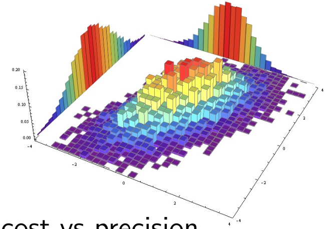
- This requires to verify
  - $Op(x)$: Probability Density Estimation, and
  - $I(x)$: the robustness of the possible inputs, which can be done with DNN verification
- based on an assumption that different inputs' local robustness are independent.
- i.e., we need to explore a partition of the input space, and use $Op(x)$ to weight the verification results of cells

Partition and Cell Size

- Binning, to partition the input space as cells
  - r-separation [Yang et al., 2020]: a data distribution over $\bigcup_{i \in C} \mathcal{X}^i$ is r-separable if for all $i, j \in C$

$$\min_{x \in \mathcal{X}^i, x' \in \mathcal{X}^j} dist(x, x') \geq 2r$$

  - use r as the radius of the cells



- need a balance between cost vs precision

How to evaluate reliability?

Table 1: The RAM details and results. For image datasets, the r, ϵ and # are associated with latent spaces. Time is in seconds per cell.

	train/test error	r-separation	cell radius ϵ	# of cells	ACU	$\mathbb{E}[\lambda]$	$\mathbb{V}[\lambda]$	$Ub_{97.5\%}$	time
The run. exp.	0.0005/0.0180	0.004013	0.004	250×250	0.002982	0.004891	0.000004	0.004899	0.04
Synth. DS-1	0.0037/0.0800	0.004392	0.004	250×250	0.008025	0.008290	0.000014	0.008319	0.03
Synth. DS-2	0.0004/0.0079	0.002001	0.002	500×500	0.004739	0.005249	0.000002	0.005252	0.04
MNIST	0.0051/0.0235	0.1003	0.100	top-170000	0.106615	0.036517	/	/	0.43
CIFAR10	0.0199/0.0853	0.1947	0.125	top-23000	0.238138	0.234419	/	/	6.74

Figure: Part of the results for Siemens' Artificial Intelligence Dependability Assessment challenge.

- How to make the estimation more precise?
  - local robustness computation cannot be speed up.
  - So, deal with networks with better generalisability! How?

Considering an important open question – **Is there any structural information that can be utilised to determine the generalisation ability of deep neural networks? [Jin et al., 2020]**

[Jin et al., 2020] (NeurIPS2020) How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks.

PAC-Bayes Bound

(McAllester, 1999) considers a generalization bound on the parameters

Posteriori distribution Q on parameters Θ

Priori distribution P on parameters Θ

$$\mathbb{E}_{\Theta\sim Q}[\mathcal{L}_D(f_\Theta)] \leq \mathbb{E}_{\Theta\sim Q}[\mathcal{L}_S(f_\Theta)] + \sqrt{\frac{\mathrm{KL}(Q||P) + \log\frac{m}{\delta}}{2(m-1)}}$$

Likelihood δ

Expected loss on input space D

Expected loss on samples S from D

Number of samples

KL divergence plays a key role in the generalization bound

- a small KL term will help tighten the bound
- a larger KL term will loose the bound

- Relax the i.i.d. assumption on the posterior distribution, consider weight correlation, in order to achieve a tighter lower bound and a better prediction ability.
- Found that there are structural components that affect the generalisability of neural networks

PAC Bayes + Weight Correlation

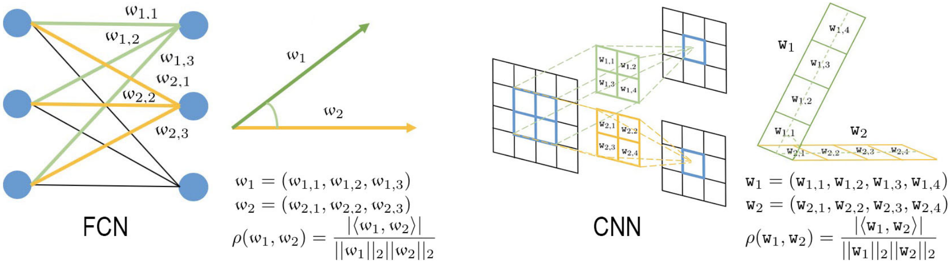


Figure: **(FCN)** WC of any two neurons is the cosine similarity of the associated weight vectors. **(CNN)** WC of any two filters is the cosine similarity of the reshaped filter matrices.

[Jin et al., 2020] (NeurIPS2020) How does Weight Correlation Affect the Generalisation Ability of Deep Neural Networks.

- Enable the design of new measure (in the next slide) which can serve as a strong/direct indicator of the generalisation. Roughly, lower weight correlation suggests a better PAC Bayes bound, i.e., smaller generalisation gap and better generalisation ability.

Table 1: Complexity Measures (Measured Quantities)

Generalisation Error (GE)	$\mathcal{L}_D(f_{\theta^F}) - \mathcal{L}_S(f_{\theta^F})$
Product of Frobenius Norms (PFN)	$\prod_\ell \lVert\theta_\ell^F\rVert_{\mathrm{Fr}}$
Product of Spectral Norms (PSN)	$\prod_\ell \lVert\theta_\ell^F\rVert_2$
Number of Parameters (NoP)	Total number of parameters in the network
Sum of Spectral Norms (SoSP)	Total number of parameters $\times \sum_\ell \lVert\theta_\ell^0 - \theta_\ell^F\rVert_2$
Weight Correlation (WC)	$\frac{1}{\ell}\sum_\ell \rho(w_\ell)$
PAC Bayes (PB)	$\sum_\ell \lVert\theta_\ell^0 - \theta_\ell^F\rVert_{\mathrm{Fr}}^2 / 2\sigma_\ell^2$
PAC Bayes & Correlation (PBC)	$\sum_\ell (\lVert\theta_\ell^0 - \theta_\ell^F\rVert_{\mathrm{Fr}}^2 / 2\sigma_\ell^2 + \mathrm{g}(w_\ell))$

New measure

Table 2: Complexity measures for CIFAR-10

Network	PFN	PSN	NoP	SoSP	PB	PBC	WC	GE
FCN1	8.1e7	1.4e4	3.7e7	1.6e9	1.1e4	1.14e5	0.297	2.056
FCN2	3.3e7	8.5e3	4.2e7	1.61e9	8.8e3	1.24e5	0.296	2.354
VGG11	8.5e10	1.4e5	9.7e6	2.4e8	2.0e3	3.41e4	0.273	0.929
VGG16	5.1e15	1.3e7	1.5e7	5.2e8	2.6e3	3.73e4	0.275	0.553
VGG19	1.1e19	2.9e8	2.1e7	8.1e8	3.3e3	4.26e4	0.274	0.678
ResNet18	2.5e22	1.1e12	1.1e7	8.4e8	4.7e3	1.34e5	0.732	2.681
ResNet34	9.9e34	4.9e16	2.1e7	3.1e9	1.0e4	1.30e5	0.733	2.552
ResNet50	1.4e76	7.5e46	2.3e7	6.1e9	1.6e7	1.62e7	0.278	2.807
DenseNet121	5.9e176	1.4e151	6.8e6	1.5e10	1.0e9	1.04e9	0.357	1.437
Concordant Pairs	21	21	22	26	24	**29**	24	-
Discordant Pairs	15	15	14	10	12	**7**	12	-
Kendall's τ	0.16	0.16	0.22	0.44	0.33	**0.61**	0.33	-

- Enable the design of new measure (in the next slide) which can serve as a strong indicator of the generalisation. Roughly, lower weight correlation suggests a better PAC Bayes bound, i.e., smaller generalisation gap and better generalisation ability.
- The training process by monitoring and adapting this measure can lead to models with better generalisation.

PAC Bayes + Weight Correlation

- Enable the design of new measure (in the next slide) which can serve as a strong indicator of the generalisation. Roughly, lower weight correlation suggests a better PAC Bayes bound, i.e., smaller generalisation gap and better generalisation ability.
- The training process by monitoring and adapting this measure can lead to models with better generalisation.
- Now close the loop: use structural information to improve the generalisation of neural network, on which reliability estimation is more accurate.

Table of Contents

Introduction

Falsification through Adversarial Attack
- Algorithms for Adversarial Attacks
- More Examples of Adversarial Attacks

Rectification through Adversarial Training
- Adversarial Training
- Distributional Robustness
- Robustness vs Generalisation

Robustness Verification
- Over-approximation Techniques
- Constraint Solving Techniques
- Abstraction and Refinement Techniques

Verification in Practice

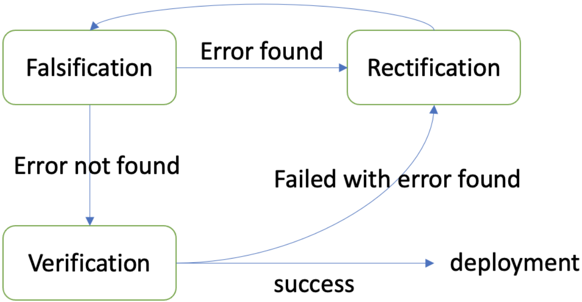
Conclusions and Future Directions

Conclusions and Future Directions

Relationship between Attacks/Defences/Verification

- Falsification through Attacks: identify risks
- Rectification through Defence: reduce risks
- Verification: prove the absence of risks

with Falsification/Rectification/Verification in mind,



- A proper metric that is of high fidelity to human perception would be key for high-quality attacks
- Attacks essentially prove the non-robustness of the model, so combining attacks (or falsification) with verification could provide a more balanced and efficient way for certified robustness evaluation
- Developing black-box attacks that the adversary can only access the hard label with limited queries
- Exploring empirical and theoretical connections between adversarial robustness and interpretability
- Exploring adversarial attacks that can resemble a wide range of real-world adversarial instances/scenarios
- Attacking solutions that are independent of a certain distance metric (or workable on multiple distance metrics)
- The empirical and theoretical relations between universal attacks and global robustness (or robustness of model structure that is independent of concrete inputs)

Future Directions – Adversarial Defence

- Theoretical understanding of adversarial training
  - What is the trade-off between robustness and accuracy?
  - How to optimally integrate both local and global information?
  - How does robustness interact with generalisation?
- Adversarial training for semi-supervised or unsupervised learning
- Adversarial training in the distributed learning scenarios, e.g., federated learning

Future Directions – Verification

- Verification for reliability (i.e., not only robustness),
- Verification of online learning,
- Improved scalability through e.g., abstraction,
- Training for verification: models that are easier to verify,
- etc.

- Ashok, P., Hashemi, V., Kretínský, J., and Mohr, S. (2020).
  Deepabstract: Neural network abstraction for accelerating verification.
  In Hung, D. V. and Sokolsky, O., editors, *Proceedings of the 18th International Symposium on Automated Technology for Verification and Analysis (ATVA20)*, volume 12302 of *Lecture Notes in Computer Science*, pages 92–107. Springer.

- Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., and Misener, R. (2020).
  Efficient verification of neural networks via dependency analysis.
  In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI20)*, pages 3291–3299. AAAI Press.

- Cheng, C., Nührenberg, G., and Ruess, H. (2017).
  Maximum resilience of artificial neural networks.
  In *International Symposium on Automated Technology for Verification and Analysis (ATVA17)*, pages 251–268. Springer.

Ehlers, R. (2017).
Formal verification of piece-wise linear feed-forward neural networks.
In *Proceedings of the 15th International Symposium on Automated Technology for Verification and Analysis (ATVA17)*, volume 10482 of *Lecture Notes in Computer Science*, pages 269–286. Springer.

Elboher, Y., Gottschlich, J., and Katz, G. (2020).
An abstraction-based framework for neural network verification.
In *Proceedings of the 32nd International Conference on Computer Aided Verification (CAV20)*, volume 12224 of *Lecture Notes in Computer Science*, pages 43–65. Springer.

Finlayson, S. G., Bowers, J. D., Ito, J., Zittrain, J. L., Beam, A. L., and Kohane, I. S. (2019).
Adversarial attacks on medical machine learning.
Science, 363(6433):1287–1289.

Finlayson, S. G., Chung, H. W., Kohane, I. S., and Beam, A. L. (2018).
Adversarial attacks against medical deep learning systems.
arXiv preprint arXiv:1804.05296.

Fischetti, M. and Jo, J. (2018).
Deep neural networks and mixed integer linear optimization.
Constraints, 23(3):296–309.

- Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. (2018).
  AI^2: Safety and robustness certification of neural networks with abstract interpretation.
  In *IEEE Symposium on Security and Privacy (S&P18)*, pages 948–963.

- Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014).
  Explaining and harnessing adversarial examples.
  arXiv preprint arXiv:1412.6572.

- Henriksen, P. and Lomuscio, A.
  DEEPSPLIT: an efficient splitting method for neural network verification via indirect effect analysis.
  In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2549–2555. ijcai.org.

Henriksen, P. and Lomuscio, A. (2020).
Efficient neural network verification via adaptive refinement and adversarial search.
In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI20)*, pages 2513–2520. IOS Press.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. (2017).
Safety verification of deep neural networks.
In *CAV2017*, pages 3–29.

Jin, G., Yi, X., Zhang, L., Zhang, L., Schewe, S., and Huang, X. (2020).
How does weight correlation affect the generalisation ability of deep neural networks.
In *NeurIPS'20*.

Katz, G., Barrett, C., Dill, D., Julian, K., and Kochenderfer, M. (2017).
Reluplex: An efficient SMT solver for verifying deep neural networks.
In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV17)*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer.

Katz, G., Huang, D., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D., Kochenderfer, M., and Barrett, C. (2019).
The marabou framework for verification and analysis of deep neural networks.
In *Proceedings of the 31st International Conference on Computer Aided Verification (CAV19)*, pages 443–452.

Kouvaros, P. and Lomuscio, A. (2021).
Towards scalable complete verification of relu neural networks via dependency-based branching.
In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2643–2650. ijcai.org.

Lomuscio, A. and Maganti, L. (2017).
An approach to reachability analysis for feed-forward relu neural networks.
arXiv preprint 1706.07351.

Pei, K., Cao, Y., Yang, J., and Jana, S. (2017).
DeepXplore: Automated Whitebox Testing of Deep Learning Systems.
In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 1–18, New York, NY, USA. Association for Computing Machinery.

Prabhakar, P. and Afzal, Z. (2019).
Abstraction based output range analysis for neural networks.
In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS19)*, pages 15762–15772.

Royo, V. R., Calandra, R., Stipanovic, D., and Tomlin, C. (2019).
Fast neural network verification via shadow prices.
CoRR, abs/1902.07247.

Ruan, W., Huang, X., and Kwiatkowska, M. (2018).
Reachability analysis of deep neural networks with provable guarantees.
In *IJCAI2018*, pages 2651–2659.

Ruan, W., Wu, M., Sun, Y., Huang, X., Kroening, D., and Kwiatkowska, M. (2019).
Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance.
In *IJCAI2019*, pages 5944–5952.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. (2019a).
Boosting robustness certification of neural networks.
In *ICLR19*. OpenReview.net.

Singh, G., Gehr, T., Püschel, M., and Vechev, P. (2019b).
An abstract domain for certifying neural networks.
In *ACM on Programming Languages*, volume 3, pages 1–30. ACM Press.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. (2018).
Fast and effective robustness certification.
In *NeurIPS18*, pages 10802–10813. Curran Associates, Inc.

Sotoudeh, M. and Thakur, A. (2020).
Abstract neural networks.
arXiv preprint 2009.05660.

Sun, Y., Huang, X., Kroening, D., Sharp, J., Hill, M., and Ashmore, R. (2019).
Structural test coverage criteria for deep neural networks.
ACM Trans. Embed. Comput. Syst., 18(5s).

Sun, Y., Wu, M., Ruan, W., Huang, X., Kwiatkowska, M., and Kroening, D. (2018).
Concolic testing for deep neural networks.
In *ASE2018*.

Tjeng, V., Xiao, K., and Tedrake, R. (2019).
Evaluating robustness of neural networks with mixed integer programming.
In *Proceedings of the 7th International Conference on Learning Representations (ICLR19)*.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018a).
Efficient formal safety analysis of neural networks.
In *NeurIPS18*, pages 6367–6377. Curran Associates, Inc.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. (2018b).
Formal security analysis of neural networks using symbolic intervals.
In *Proceedings of the 27th USENIX Security Symposium (USENIX18)*.

Webb, S., Rainforth, T., Teh, Y. W., and Kumar, M. P. (2019).
A statistical approach to assessing neural network robustness.
In *ICLR2019*.

Weng, T., Zhang, H., Chen, H., Song, Z., Hsieh, C., Boning, D., Dhillon, I., and Daniel, L. (2018).
Towards fast computation of certified robustness for relu networks.
In *Proceedings of the 35th International Conference on Machine Learning (ICML18)*.

Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. (2018).
Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach.
In *ICLR2018*.

Wicker, M., Huang, X., and Kwiatkowska, M. (2018).
Feature-guided black-box safety testing of deep neural networks.
In *TACAS2018*, pages 408–426.

Wu, M., Wicker, M., Ruan, W., Huang, X., and Kwiatkowska, M. (2020).
A game-based approximate verification of deep neural networks with provable guarantees.
Theoretical Computer Science, 807:298–329.
In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part II.

Wu, Y. and Zhang, M. (2021).
Tightening robustness verification of convolutional neural networks with fine-grained linear approximation.
In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI21)*.
AAAI Press.

References XIV

Yang, Y.-Y., Rashtchian, C., Zhang, H., Salakhutdinov, R. R., and Chaudhuri, K. (2020).
A closer look at accuracy vs. robustness.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8588–8601. Curran Associates, Inc.

Zhang, H., Weng, T., Chen, P., Hsieh, C., and Daniel, L. (2018).
Efficient neural network robustness certification with general activation functions.
In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems 2018 (NeurIPS18)*, pages 4944–4953.

Zhao, X., Banks, A., Sharp, J., Robu, V., Flynn, D., Fisher, M., and Huang, X. (2020).
A safety framework for critical systems utilising deep neural networks.
In *SafeComp2020*, pages 244–259.

Zhao, X., Huang, W., Banks, A., Cox, V., Flynn, D., Schewe, S., and Huang, X. (2021).
Assessing the reliability of deep learning classifiers through robustness evaluation and operational profiles.
In *AISafety2021*.