Robustness Certificates for Implicit Neural Networks:

A Mixed Monotone Contractive Approach

Saber Jafarpour [1,*], Matthew Abate [1,*], Alexander Davydov [2,*], Francesco Bullo [2], and Samuel Coogan [1]

(1) Decision and Control Laboratory
Georgia Institute of Technology

(2) Center for Control, Dynamical Systems, and Computation
University of California, Santa Barbara

- Increase in computational power of neural networks
- **However**, neural networks can be fragile wrt input perturbations

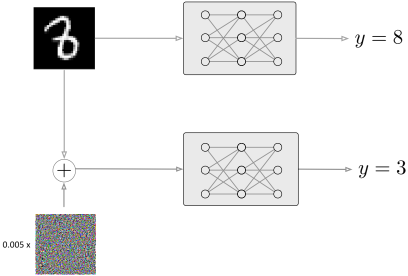
- Increase in computational power of neural networks
- **However**, neural networks can be fragile wrt input perturbations

Adversarial Perturbations

Small changes in the input

↓

Large changes in the output



C. Szegedy and et. al. Intriguing properties of neural networks. In *ICLR*, 2014

- Robustness of neural networks is critical in their real-world applications

Neural Networks

Verification and Training

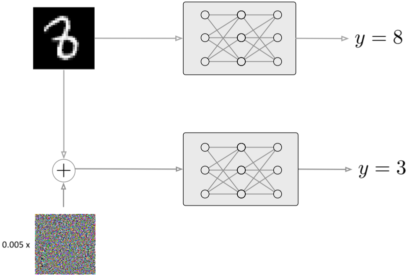
- Increase in computational power of neural networks
- **However**, neural networks can be fragile wrt input perturbations

Adversarial Perturbations

Small changes in the input

$\downarrow$

Large changes in the output



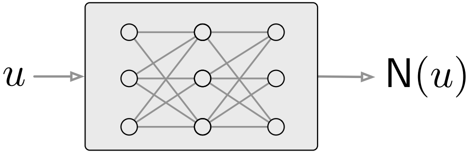
C. Szegedy and et. al. Intriguing properties of neural networks. In *ICLR*, 2014

- Robustness of neural networks is critical in their real-world applications

1. **Verification**: how robust is a given neural network?
2. **Training**: how to design robust neural networks?

Given an input perturbation set $\mathcal{U}$
Safe output domain $\mathcal{S}$

$$\mathsf{N}(\mathcal{U}) = \{\mathsf{N}(u) \mid u \in \mathcal{U}\}$$



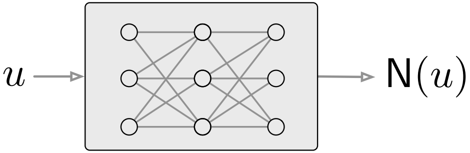
Goal: over-approximate $\mathsf{N}(\mathcal{U})$ and check if $\mathsf{N}(\mathcal{U}) \subset \mathcal{S}$.

Robustness Analysis

A paradigm for safety verification

Given an input perturbation set $\mathcal{U}$
Safe output domain $\mathcal{S}$

$$\mathsf{N}(\mathcal{U}) = \{\mathsf{N}(u) \mid u \in \mathcal{U}\}$$



Goal: over-approximate $\mathsf{N}(\mathcal{U})$ and check if $\mathsf{N}(\mathcal{U}) \subset \mathcal{S}$.

- **Lipschitz estimates:**

  A. Virmaux and K. Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *NeurIPS*, 2018

- **Interval arithmetic:**

  W. Xiang, H.-D. Tran, and T. T. Johnson. Output reachable set estimation and verification for multilayer neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 2018

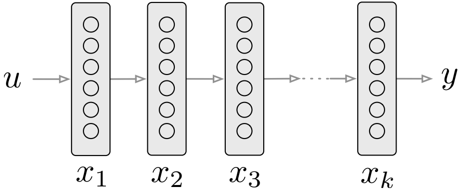
- **Semi-definite programing:**

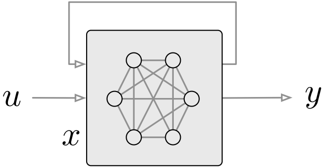
  M. Fazlyab, M. Morari, and G. J. Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2020.

Implicit Neural Networks (INNs)

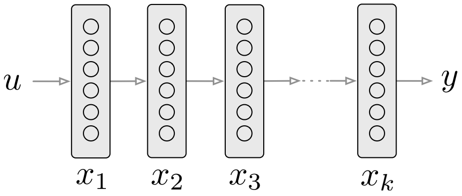
Definition via fixed-point equations

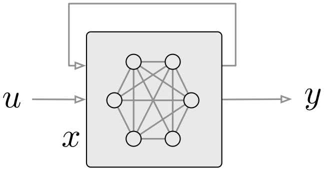
- explicit hidden layers are replaced by a single implicit layer





- explicit hidden layers are replaced by a single implicit layer





- traditional neural networks:

$$x^{i+1} = \Phi(A_i x^i + b_i), \quad x^0 = u$$
$$y = A_k x^k + b_k$$

- implicit neural networks:

$$x = \Phi(Ax + Bu + b)$$
$$y = Cx + c$$

- $\Phi(y_1, \dots, y_n) = (\phi_1(y_1), \dots, \phi_n(y_n))^\top$ is a diagonal activation function
- activation functions are slope-restricted in $[0, 1]$, i.e., $0 \leq \frac{\phi_i(x)-\phi_i(y)}{x-y} \leq 1$ for all $x, y \in \mathbb{R}$

Notion of Layer: output is defined **implicitly** as a function of input

e.g., fixed-point equation, differential equations, optimization problem

Origins

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *NeurIPS*, 2019

L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Y. Tsai. Implicit deep learning. *SIMODS*, 2019

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017

Notion of Layer: output is defined **implicitly** as a function of input

e.g., fixed-point equation, differential equations, optimization problem

Origins

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *NeurIPS*, 2019

L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Y. Tsai. Implicit deep learning. *SIMODS*, 2019

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018

B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017

Motivations for using implicit learning:

- **Representation**: a general class of learning models
  - includes feedforward and residual neural networks
  - architecture flexibility
- **Performance**: differential equations and optimization problems
- **Memory**: comparable accuracy to traditional networks with significant memory reduction

Implicit Neural Networks (INNs)

A dynamical system perspective

- **Challenge 1**: well-posedness, i.e., existence of solutions to $x = \Phi(Ax + Bu + b)$
- **Challenge 2**: computing robustness margin, i.e., over-approximating $\mathsf{N}(\mathcal{U})$ (N input-output map)

Implicit Neural Networks (INNs)

A dynamical system perspective

- **Challenge 1**: well-posedness, i.e., existence of solutions to $x = \Phi(Ax + Bu + b)$
- **Challenge 2**: computing robustness margin, i.e., over-approximating $\mathsf{N}(\mathcal{U})$ (N input-output map)

Key insight

Fixed-point equation $x = \Phi(Ax + Bu + b)$	$\Longleftrightarrow$	Dynamical system $\dot{x} = -x + \Phi(Ax + Bu + b)$
well-posedness	$\Longleftrightarrow$	**equilibrium points**
robustness	$\Longleftrightarrow$	**forward reachability**

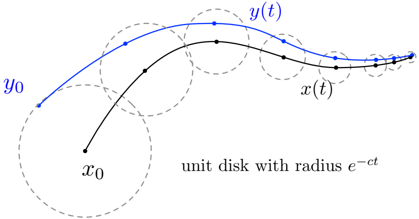
- Tools and techniques from dynamical system and control theory
- We use Contraction Theory and Mixed Monotone System Theory

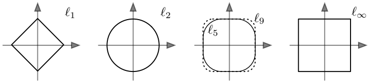
Aside #1: Contraction Theory

A framework for well-posedness

Definition

$\dot{x} = \mathsf{F}(t,x)$ is contracting wrt $\|\cdot\|$ if its flow is a contraction map wrt $\|\cdot\|$





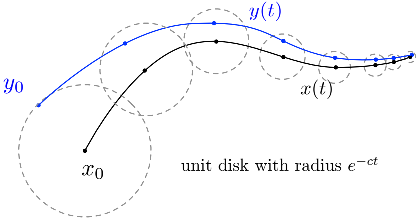
Definition

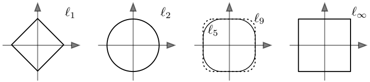
$\dot{x} = \mathsf{F}(t,x)$ is contracting wrt $\|\cdot\|$ if its flow is a contraction map wrt $\|\cdot\|$

Contraction via Logarithmic norms

$\dot{x} = \mathsf{F}(t,x)$ is contracting wrt $\|\cdot\|$ with rate c iff

$$\mu_{\|\cdot\|}(D\mathsf{F}(t,x)) \leq -c, \qquad \text{for all } t, x$$





- **logarithmic norm** $\mu_{\|\cdot\|}(A) := \lim_{h\to 0^+} \frac{\|I_n + hA\| - 1}{h}$
- Formula: $\mu_2(A) = \frac{1}{2}\lambda_{\max}(A + A^\top)$

$$\mu_1(A) = \max_j \left(a_{jj} + \sum_{i\neq j} |a_{ij}|\right)$$

$$\mu_\infty(A) = \max_i \left(a_{ii} + \sum_{j\neq i} |a_{ij}|\right)$$

Aside #1: Contraction Theory

A framework for well-posedness

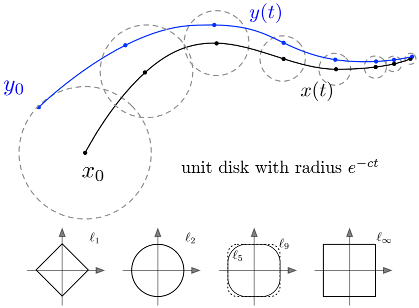
Definition

$\dot{x} = \mathsf{F}(t,x)$ is contracting wrt $\|\cdot\|$ if its flow is a contraction map wrt $\|\cdot\|$

Contraction via Logarithmic norms

$\dot{x} = \mathsf{F}(t,x)$ is contracting wrt $\|\cdot\|$ with rate c iff

$$\mu_{\|\cdot\|}(D\mathsf{F}(t,x)) \leq -c, \qquad \text{for all } t, x$$



- **logarithmic norm** $\mu_{\|\cdot\|}(A) := \lim_{h \to 0^+} \frac{\|I_n + hA\| - 1}{h}$
- Formula: $\mu_2(A) = \frac{1}{2}\lambda_{\max}(A + A^\top)$

$$\mu_1(A) = \max_j \left(a_{jj} + \sum_{i \neq j} |a_{ij}|\right)$$

$$\mu_\infty(A) = \max_i \left(a_{ii} + \sum_{j \neq i} |a_{ij}|\right)$$

A contracting $\dot{x} = \mathsf{F}(x)$ (**time-invariant**) converges to a unique equilibrium point

Aside #2: Mixed Monotone System Theory

A framework for reachability analysis

Original system

$$\dot{x} = \mathsf{F}(x, u)$$

Embedded system

$$\dot{\underline{x}} = \mathsf{G}(\underline{x}, \overline{x}, \underline{u}, \overline{u}),$$
$$\dot{\overline{x}} = \mathsf{G}(\overline{x}, \underline{x}, \overline{u}, \underline{u})$$

1. F is embedded in G, i.e., $\mathsf{F}(x, u) = \mathsf{G}(x, x, u, u)$
2. D_1G is Metzler and D_2G is non-positive
3. D_3G is non-negative and D_4G is non-positive

- Metzler = non-negative off-diagonal entries
- embedded system is a monotone dynamical system wrt the **southeast order**

$$\begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{y} \\ \overline{y} \end{bmatrix} \quad \Longleftrightarrow \quad \underline{x} \leq \underline{y}, \ \ \overline{y} \leq \overline{x}$$

- G is not unique and different approaches exist for computing G

Original system

$$\dot{x} = \mathsf{F}(x, u)$$

Embedded system

$$\dot{\underline{x}} = \mathsf{G}(\underline{x}, \overline{x}, \underline{u}, \overline{u}),$$
$$\dot{\overline{x}} = \mathsf{G}(\overline{x}, \underline{x}, \overline{u}, \underline{u})$$

1. F is embedded in G, i.e., $\mathsf{F}(x, u) = \mathsf{G}(x, x, u, u)$
2. $D_1\mathsf{G}$ is Metzler and $D_2\mathsf{G}$ is non-positive
3. $D_3\mathsf{G}$ is non-negative and $D_4\mathsf{G}$ is non-positive

Reachability via embedded system

For $u \in [\underline{u}, \overline{u}]$, every trajectory of F starting from $x_0 \in [\underline{x}_0, \overline{x}_0]$ satisfies

$$x(t) \in [\underline{x}(t), \overline{x}(t)]$$

where $t \mapsto \begin{bmatrix} \underline{x}(t) \\ \overline{x}(t) \end{bmatrix}$ is the trajectory of embedded system starting from $\begin{bmatrix} \underline{x}_0 \\ \overline{x}_0 \end{bmatrix}$

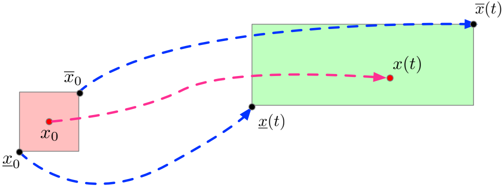
- Metzler = non-negative off-diagonal entries
- embedded system is a monotone dynamical system wrt the **southeast order**

$$\begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} \leq_{\mathrm{SE}} \begin{bmatrix} \underline{y} \\ \overline{y} \end{bmatrix} \iff \underline{x} \leq \underline{y}, \ \ \overline{y} \leq \overline{x}$$

- G is not unique and different approaches exist for computing G

Blue = embedded system

Red = original system



- **Metzler/non-Metzler** decomposition: $A = \lceil A \rceil^{\mathrm{Mzl}} + \lfloor A \rfloor^{\mathrm{Mzl}}$
- Example: $A = \begin{bmatrix} 2 & -1 \\ 1 & -3 \end{bmatrix} \implies \lceil A \rceil^{\mathrm{Mzl}} = \begin{bmatrix} 2 & 0 \\ 1 & -3 \end{bmatrix} \quad \lfloor A \rfloor^{\mathrm{Mzl}} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$

- **Metzler/non-Metzler** decomposition: $A = \lceil A \rceil^{\mathrm{Mzl}} + \lfloor A \rfloor^{\mathrm{Mzl}}$
- Example: $A = \begin{bmatrix} 2 & -1 \\ 1 & -3 \end{bmatrix} \implies \lceil A \rceil^{\mathrm{Mzl}} = \begin{bmatrix} 2 & 0 \\ 1 & -3 \end{bmatrix} \quad \lfloor A \rfloor^{\mathrm{Mzl}} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$

Dynamical system perspective

Original system $u \in [\underline{u}, \overline{u}]$

$$\dot{x} = -x + \Phi(Ax + Bu + b)$$

$\implies$

Embedded system

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\overline{x}} \end{bmatrix} = - \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} \Phi(\lceil A \rceil^{\mathrm{Mzl}} \underline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \overline{x} + [B]^{+} \underline{u} + [B]^{-} \overline{u} + b) \\ \Phi(\lceil A \rceil^{\mathrm{Mzl}} \overline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \underline{x} + [B]^{+} \overline{u} + [B]^{-} \underline{u} + b) \end{bmatrix}$$

- **Metzler/non-Metzler** decomposition: $A = \lceil A \rceil^{\mathrm{Mzl}} + \lfloor A \rfloor^{\mathrm{Mzl}}$
- Example: $A = \begin{bmatrix} 2 & -1 \\ 1 & -3 \end{bmatrix} \implies \lceil A \rceil^{\mathrm{Mzl}} = \begin{bmatrix} 2 & 0 \\ 1 & -3 \end{bmatrix} \quad \lfloor A \rfloor^{\mathrm{Mzl}} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$

Dynamical system perspective

Original system $u \in [\underline{u}, \overline{u}]$

$$\dot{x} = -x + \Phi(Ax + Bu + b) \implies$$

Embedded system

$$\begin{bmatrix} \dot{\underline{x}} \\ \dot{\overline{x}} \end{bmatrix} = - \begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} + \begin{bmatrix} \Phi(\lceil A \rceil^{\mathrm{Mzl}} \underline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \overline{x} + [B]^{+} \underline{u} + [B]^{-} \overline{u} + b) \\ \Phi(\lceil A \rceil^{\mathrm{Mzl}} \overline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \underline{x} + [B]^{+} \overline{u} + [B]^{-} \underline{u} + b) \end{bmatrix}$$

Fixed-point equation perspective

Original INN $u \in [\underline{u}, \overline{u}]$

$$x = \Phi(Ax + Bu + b) \implies$$

Embedded INN

$$\begin{bmatrix} \underline{x} \\ \overline{x} \end{bmatrix} = \begin{bmatrix} \Phi(\lceil A \rceil^{\mathrm{Mzl}} \underline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \overline{x} + [B]^{+} \underline{u} + [B]^{-} \overline{u} + b) \\ \Phi(\lceil A \rceil^{\mathrm{Mzl}} \overline{x} + \lfloor A \rfloor^{\mathrm{Mzl}} \underline{x} + [B]^{+} \overline{u} + [B]^{-} \underline{u} + b) \end{bmatrix}$$

Embedded INN

Main result

Theorem

If $\mu_\infty(A) < 1$ and $u \in [\underline{u}, \overline{u}]$

1. INN has a unique fixed point x^*_u
2. Embedded INN has a unique fixed point $\begin{bmatrix} \underline{x}^* \\ \overline{x}^* \end{bmatrix}$
3. $\underbrace{\left([C]^+ \ [C]^-\right) \begin{bmatrix} \underline{x}^* \\ \overline{x}^* \end{bmatrix} + c}_{\underline{y}} \leq y \leq \underbrace{\left([C]^- \ [C]^+\right) \begin{bmatrix} \underline{x}^* \\ \overline{x}^* \end{bmatrix} + c}_{\overline{y}}$

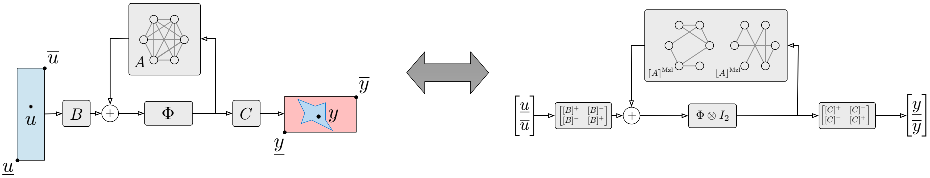
Embedded INN

Main result

Theorem

If $\mu_\infty(A) < 1$ and $u \in [\underline{u}, \overline{u}]$

1. INN has a unique fixed point x_u^*
2. Embedded INN has a unique fixed point $\begin{bmatrix} \underline{x}^* \\ \overline{x}^* \end{bmatrix}$
3. $\underbrace{\left([C]^+ \ [C]^-\right) \begin{bmatrix} \underline{x}^* \\ \overline{x}^* \end{bmatrix}}_{\underline{y}} + c \leq y \leq \underbrace{\left([C]^- \ [C]^+\right) \begin{bmatrix} \underline{x}^* \\ \overline{x}^* \end{bmatrix}}_{\overline{y}} + c$



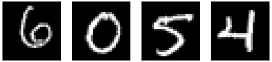
- Generalization of Interval Bound Propagation (IBP) approach

S. Gowal and et. al. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint*, 2018

Numerical Experiments

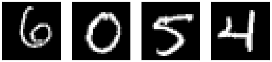
MNIST dataset classification

- MNIST dataset: 28×28 pixel handwritten digits between $0 - 9$.
- INN with $n = 100$ and trained using NEMON algorithm*
  - $\epsilon =$ size of perturbation, $\mathcal{U} = [u - \epsilon \mathbb{1}_{784}, u + \epsilon \mathbb{1}_{784}]$.



*S. Jafarpour, A. Davydov, A. V. Proskurnikov, and F. Bullo. Robust implicit networks via non-Euclidean contractions. In *NeurIPS*, Dec. 2021

- MNIST dataset: 28×28 pixel handwritten digits between $0 - 9$.
- INN with $n = 100$ and trained using NEMON algorithm*
  - $\epsilon =$ size of perturbation, $\mathcal{U} = [u - \epsilon \mathbb{1}_{784}, u + \epsilon \mathbb{1}_{784}]$.



Lipschitz Approach*

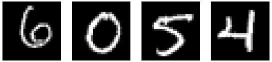
$$\mathsf{N}(\mathcal{U}) \subset [y - \mathsf{Lip}_\infty \epsilon, y + \mathsf{Lip}_\infty \epsilon]$$

Mixed Monotone Approach

$$\mathsf{N}(\mathcal{U}) \subset [\underline{y}(\epsilon), \overline{y}(\epsilon)]$$

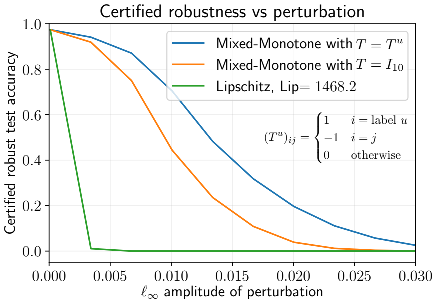
* S. Jafarpour, A. Davydov, A. V. Proskurnikov, and F. Bullo. Robust implicit networks via non-Euclidean contractions. In *NeurIPS*, Dec. 2021

- MNIST dataset: 28×28 pixel handwritten digits between $0-9$.
- INN with $n = 100$ and trained using NEMON algorithm*
- $\epsilon =$ size of perturbation, $\mathcal{U} = [u - \epsilon \mathbb{1}_{784}, u + \epsilon \mathbb{1}_{784}]$.



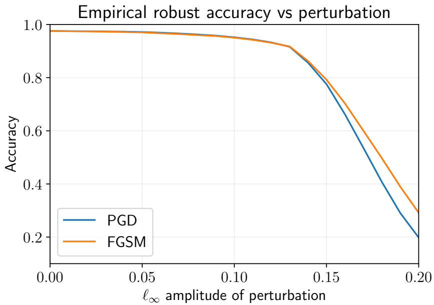
Lipschitz Approach*

$$\mathsf{N}(\mathcal{U}) \subset [y - \mathsf{Lip}_\infty \epsilon, y + \mathsf{Lip}_\infty \epsilon]$$



Mixed Monotone Approach

$$\mathsf{N}(\mathcal{U}) \subset [\underline{y}(\epsilon), \overline{y}(\epsilon)]$$



* S. Jafarpour, A. Davydov, A. V. Proskurnikov, and F. Bullo. Robust implicit networks via non-Euclidean contractions. In *NeurIPS*, Dec. 2021

Conclusions

- A dynamical system perspective to robustness analysis of neural network
- Contraction theory + Mixed monotone system theory
- Hyper-rectangular over-approximation of reachable sets of INNs

Conclusions

- A dynamical system perspective to robustness analysis of neural network
- Contraction theory + Mixed monotone system theory
- Hyper-rectangular over-approximation of reachable sets of INNs

Future Directions

- Training robust implicit neural networks using mixed monotone theory (submitted to CDC)
- Reachability analysis of closed-loop systems with neural network controllers

Thank you for your attention!

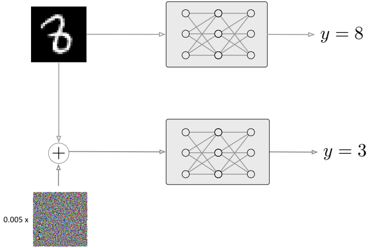
Backup slides

Adversarial perturbations

Features and mitigation

Feature of adversarial perturbations:

- exist for a large class of learning algorithms
- transfer across models (not always!)
- *not* caused by overfitting (empirical evidence)



How to mitigate the effect of adversarial perturbations?

Adversarial training

- improve training using an attack
- easy to implement
- no provable guarantee

Robust optimization

- use over-approximation of the output
- hard to implement in training
- provide guarantees

- A large and flexible class of neural networks: includes feedforward neural networks

$$x^{i+1} = \Phi(A_i x^i + b_i), \qquad \text{for all } i \in \{0, \dots, k-1\}$$
$$y = A_k x^k + b_k, \quad u = x^0$$

The equivalent INN is given by:

$$\begin{bmatrix} x^k \\ x^{k-1} \\ \vdots \\ x^2 \\ x^1 \end{bmatrix} = \Phi\left(\begin{bmatrix} 0 & A_{k-1} & 0 & \dots & 0 \\ 0 & 0 & A_{k-2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & A_1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} x^k \\ x^{k-1} \\ \vdots \\ x^2 \\ x^1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ A_0 \end{bmatrix} u + \begin{bmatrix} b_{k-1} \\ b_{k-2} \\ \vdots \\ b_1 \\ b_0 \end{bmatrix} \right),$$

$$y = \begin{bmatrix} A_k & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} x^k \\ x^{k-1} \\ \vdots \\ x^2 \\ x^1 \end{bmatrix} + b_k$$

1. **Origins**

S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *NeurIPS*, 2019

L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Y. Tsai. Implicit deep learning. *SIMODS*, 2019

A. Kag, Z. Zhang, and V. Saligrama. RNNs incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In *ICLR*, 2020

2. **Monotone operator theory**

E. Winston and J. Z. Kolter. Monotone operator equilibrium networks. In *NeurIPS*, 2020

M. Revay, R. Wang, and I. R. Manchester. Lipschitz bounded equilibrium networks. 2020. URL https://arxiv.org/abs/2010.01732

3. **Convergence**

K. Kawaguchi. On the theory of implicit deep learning: Global convergence with implicit layers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=p-NZIuwqhI4

S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin. Fixed point networks: Implicit depth models with Jacobian-free backprop, 2021. URL https://arxiv.org/abs/2103.12803. ArXiv e-print

- Training INNs:

  1. loss function $\mathcal{L}$
  2. training data $(\widehat{u}_i, \widehat{y}_i)_{i=1}^N$
  3. **training optimization problem**

$$\min_{A,B,b,c} \quad \sum_{i=1}^{N} \mathcal{L}(\widehat{y}_i, Cx_i + c)$$
$$x_i = \Phi(Ax_i + B\widehat{u}_i + b)$$

- Efficient back-propagation through implicit differentiation
- Stochastic gradient descent: at each step solve $x = \Phi(Ax + Bu + b)$.