Safety and Robustness in Deep Learning using Semidefinite Programming

Mahyar Fazlyab, Manfred Morari, George J. Pappas
Department of Electrical and Systems Engineering
University of Pennsylvania
pappasg@seas.upenn.edu
Thank you!
Tremendous progress in deep learning for perception

Whelan, Leutenegger, Salas-Moreno, Glocker, Davison, RSS’15

Geiger, Lenz, Urtasun, CVPR’12

Newcombe, Fox, Seitz, CVPR’15

Microsoft Ignite, 2015
What about safety critical applications?

**Surprise! 2020 Is Not the Year for Self-Driving Cars**

The AV industry has had to reset expectations as it shifts its focus to Level 4 autonomy

*By Mark Anderson*

April 22, 2020

“all the technology developers are realizing that this is going to be a much more incremental process”
Challenge: Robust deep learning for safety critical systems

Change in lighting
(DeepXplore SOSP’17)

Physical attacks
(EEF+’15)

Imperceptible adversarial noise
(GSS’15)
Fragility of deep networks

\[ \begin{align*}
\Phi (y^0, x^1) &\quad \Phi (x^\ell) \\
\Phi (y^0, x^1) &\quad \Phi (x^\ell)
\end{align*} \]

\[ \text{SPEED LIMIT 40} \]
Feedforward deep networks

\[ x^{k+1} = \phi(W^k x^k + b^k) \quad k = 0, \cdots, \ell - 1 \]

dimension: \( n_x \) \quad total # of neurons: \( n \) \quad dimension: \( n_y \)

activation layer \quad linear layer \quad # of layers

ReLU \quad tanh
Robustness of deep networks

- Robust and adversarial training

\[
\min_{\theta} \mathbb{E}_{(x^*, y^*)} \left[ \max_{\delta \in \Delta} \text{loss}(f_\theta(x^* + \delta), y^*) \right]
\]

- Post training verification of deep networks

\[\mathcal{X} \rightarrow f(.) \rightarrow \mathcal{Y} := f(\mathcal{X})\]
Approaches for verifying neural networks

- **Exact** methods: calculate the output set exactly
  - Exponential run-time (only for ReLU networks)
  - **Mixed-Integer Linear Programming (MILP)** [Fischetti and Jo, 2017, Dutta et al., 2018, Tjeng, et al., 2017]

- **Inexact** methods: outer approximate the output set
  - Polynomial run-time
  - **Reachability-based-approaches** [Gehr et al, 2018, Xiang et al, 2018]
• A deep network is a linear “time-varying” system with nonlinear feedback

*J. Seidman et al, Robust Deep Learning as Optimal Control, Learning for Dynamics and Control (L4DC), June 2020
Our Approach – Deep networks as dynamical systems

• A deep network is a linear “time-varying” system with nonlinear feedback*

*J. Seidman et al, Robust Deep Learning as Optimal Control, Learning for Dynamics and Control (L4DC), June 2020
A Robust Control Approach

Our Key Idea:
• Abstract $\phi$ by quadratic constraints
• Inspired from Robust Control [Zames & Falb, ’68, Megretski & Rantzer, ‘97]
Nonlinear activation functions

- **Sigmoid**
  \[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

- **tanh**
  \[ \tanh(x) \]

- **ReLU**
  \[ \max(0, x) \]

- **Leaky ReLU**
  \[ \max(0.1x, x) \]

- **Maxout**
  \[ \max(w_1^T x + b_1, w_2^T x + b_2) \]

- **ELU**
  \[ \begin{cases} 
  x & x \geq 0 \\
  \alpha(e^x - 1) & x < 0 
  \end{cases} \]
### Definition: Slope-Restricted Nonlinearity [Zames, ’68]

We say that $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ is slope-restricted in the sector $[\alpha, \beta]$ if
\[
\alpha \leq \varphi(x_2) - \varphi(x_1) \leq \beta \quad \text{or equivalently}
\]
\[
\begin{bmatrix}
    x_1 - x_2 \\
    \varphi(x_1) - \varphi(x_2)
\end{bmatrix}^\top
\begin{bmatrix}
    -2\alpha\beta & \alpha + \beta \\
    \alpha + \beta & -2
\end{bmatrix}
\begin{bmatrix}
    x_1 - x_2 \\
    \varphi(x_1) - \varphi(x_2)
\end{bmatrix} \geq 0
\]

- All commonly used activation functions are slope restricted
ReLU abstraction by Quadratic Constraints

- Single ReLU activation function:

\[ y = \max(0, x) \quad y^2 = xy \quad y \geq 0 \quad y \geq x \]

- Relaxation: for any \( \lambda \) and any \( \nu, \eta, \mu \geq 0 \)

\[
\lambda(y^2 - xy) + \nu(y - x) + \eta y \geq 0
\]

\[
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & \lambda & -\nu \\ \lambda & -2\lambda & \nu + \eta \\ -\nu & \nu + \eta & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \geq 0.
\]
• **Our Key Idea:** Overapproximate the activation functions by quadratic constraints

![Original NN](image1)

![QCLN](image2)

Transform

Certificate for the original NN

Verify via semidefinite programming

• **Our Key Insight:** Many deep network properties (safety, robustness, local Lipschitz, global Lipschitz, detection confidence, closed loop stability) that we can prove for the abstracted network will transfer to original deep network

• **Our Main Result:** We can verify QCLN using semidefinite programming
Robustness certification using global Lipschitz constant

\[ \| x - x^* \|_2 \leq \epsilon \]

\[ \| f(x) - f(x^*) \|_2 \leq L_2 \| x - x^* \|_2 \]

\[ \| f(x) - f(x^*) \|_2 \leq \delta \]

Why important: tight upper bound on \( L_2 \) useful in
- Robustness certification of classifiers
- Closed-loop stability analysis of systems with neural network controllers
- Robust training
- Generalization bounds

Challenge: finding \( L_2 \) is NP-hard

Images at same \( \ell_2 \) distance

(Wang et al., NYU 2004)

Lower Lipschitz constant implies more robustness
Incrementally-Quadratically-Constrained Linear Network

- **Key Insight:** any property we can prove for IQCLN, holds for the original network as well

M. Fazlyab et al, Efficient and Accurate Estimation of Lipschitz Constants for DNNs, NeurIPS 2019, (Oral Spotlight)
Consider a single-layer neural network described by \( f(x) = W^1 \phi(W^0 x + b^0) + b^1 \). Suppose \( \phi(x) = [\varphi(x_1) \cdots \varphi(x_n)] \), where \( \varphi \) is slope-restricted on \([\alpha, \beta]\). Suppose there exists a \( \rho > 0 \) such that the matrix inequality

\[
M(\rho, T) := \begin{bmatrix}
-2\alpha\beta W^0^\top T W^0 - \rho I_{n_0} & (\alpha + \beta) W^0^\top T \\
(\alpha + \beta) T W^0 & -2T + W^1^\top W^1
\end{bmatrix} \preceq 0,
\]

holds for some \( T \in \mathcal{T}_n \). Then \( \|f(x) - f(y)\|_2 \leq \sqrt{\rho} \|x - y\|_2 \) for all \( x, y \in \mathbb{R}^{n_0} \).

\[
\minimize_{(\rho, T)} \rho \quad \text{subject to} \quad M(\rho, T) \preceq 0, \ T \in \mathcal{T}_n
\]

A semidefinite program for finding the best upper bound.
Comparisons with state-of-the-art: Accuracy

- **Platform:** MATLAB, CVX toolbox, and MOSEK on a 9-core CPU with 16GB of RAM
- **Methods:**

  - Our bounds are the tightest in the literature
Comparisons with state-of-the-art: Efficiency

<table>
<thead>
<tr>
<th>$n$</th>
<th>LipSDP-Neuron</th>
<th>LipSDP-Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>5.22</td>
<td>2.85</td>
</tr>
<tr>
<td>1000</td>
<td>27.91</td>
<td>17.88</td>
</tr>
<tr>
<td>1500</td>
<td>82.12</td>
<td>58.61</td>
</tr>
<tr>
<td>2000</td>
<td>200.88</td>
<td>146.09</td>
</tr>
<tr>
<td>2500</td>
<td>376.07</td>
<td>245.94</td>
</tr>
<tr>
<td>3000</td>
<td>734.63</td>
<td>473.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\ell$</th>
<th>LipSDP-Neuron</th>
<th>LipSDP-Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>20.33</td>
<td>3.41</td>
</tr>
<tr>
<td>10</td>
<td>32.18</td>
<td>7.06</td>
</tr>
<tr>
<td>50</td>
<td>87.45</td>
<td>25.88</td>
</tr>
<tr>
<td>100</td>
<td>135.85</td>
<td>40.39</td>
</tr>
<tr>
<td>200</td>
<td>221.2</td>
<td>64.90</td>
</tr>
<tr>
<td>500</td>
<td>707.56</td>
<td>216.49</td>
</tr>
</tbody>
</table>

**Table:** Computation time in seconds for evaluating Lipschitz bounds of one-hidden-layer neural networks with a varying number of hidden units. A plot showing the Lipschitz constant for each network tested in this table has been provided in the Appendix.

**Table:** Computation time in seconds for computing Lipschitz bounds of $\ell$-hidden-layer neural networks with 100 activation functions per layer. For LipSDP-Neuron and LipSDP-Layer, we split each network up into 5-layer sub-networks.
Impact of Robust Training (MNIST)

- **Robust Training Methods:**
Probabilistic Verification of Neural Networks

- Input uncertainty can be random and potentially unbounded

\[ \Pr(f(x) \in S_y) = ? \]

- **Our approach:** an SDP to propagate confidence ellipsoids [Fazlyab et al., CDC, 2019]

\[
\begin{align*}
\text{minimize} & \quad \text{Volume}(S_y) \\
\text{subject to} & \quad f(E_p) \subseteq S_y \\
\Pr(f(x) \in S_y) & \geq \Pr(f^{QCLN}(x; Q) \in S_y).
\end{align*}
\]
System Safety with Deep Networks in Feedback Loop

- Safety of closed loop systems with neural network feedback


What’s Next?

Robust learning meets robust control

- Scalability of SDP for deep learning
- Safety and stability of deep reinforcement learning
- Model-based robust deep learning (soon on arxiv)
- Robust optimal control for training of robust learning

Model-Based Robust Deep Learning

(a) Domain A.

(b) Domain B.

(c) Test domain.

(d) Results

Figure 20: GTSRB Brightness.