Formal Verification of Machine Learning

MPRI 2-6: Abstract Interpretation, Application to Verification and Static Analysis

Caterina Urban
February 7th, 2022
Year 2021-2022
Machine Learning Revolution

Computer software able to efficiently and **autonomously perform tasks** that are difficult or even **impossible** to design using explicit programming.

Examples: object recognition, image classification, speech recognition, etc.
ML in Safety-Critical Applications

Enables **new functions that could not be envisioned before**

- Self-Driving Cars
- Image-Based Taxiing, Takeoff, Landing
- Aircraft Voice Control
ML in Safety-Critical Applications

Approximates complex systems and automates decision-making

Deep Neural Network Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian\(^1\) and Mykel J. Kochenderfer\(^2\) and Michael P. Owen\(^3\)

Abstract—One approach to designing decision making logic for an aircraft collision avoidance system frames the problem as a Markov decision process and optimizes the system using dynamic programming. The resulting collision avoidance strategy can be represented as a numeric table. This methodology has been used in the development of the Airborne Collision Avoidance System X (ACAS X) family of collision avoidance systems for manned and unmanned aircraft, but the high dimensionality of the state space leads to extremely large score tables. To improve storage efficiency, a deep neural network is used to approximate the table. With the use of an asymmetric loss function and a gradient descent algorithm, the parameters for this network can be trained to provide accurate estimates of table values while preserving the relative preferences of the possible advisories for each state. By training multiple networks to represent subtables, the network also decreases the number of table entries that need to be stored, reducing the required storage space.

Diagnosis and Drug Discovery

Aircraft Collision Avoidance

Formal Verification of Machine Learning

Lesson 15

Caterina Urban
ML in Safety-Critical Applications

IBM’s Watson supercomputer recommended ‘unsafe and incorrect’ cancer treatments, internal documents show

By Casey Ross and Ike Swetlitz

July 25, 2018

A self-driving Uber ran a red light last December, contrary to company claims

Internal documents reveal that the car was at fault

By Andrew Liptak

Feb 25, 2017, 11:08am EST

Feds Say Self-Driving Uber SUV Did Not Recognize Jaywalking Pedestrian In Fatal Crash

Richard Gonzales

November 7, 2019 10:57 PM ET
Machine Learning Pipeline

- data
- data preparation
- model training
- model deployment
- predictions
Machine Learning Pipeline

Model Training is **Highly Non-Deterministic**

Model Training is **Highly Non-Deterministic**

- **data preparation**
- **model training**
- **model deployment**
- **predictions**

---

**This is your machine learning system?**

YUP! You pour the data into this big pile of linear algebra, then collect the answers on the other side. What if the answers are wrong?

Just stir the pile until they start looking right.

---

**no predictability and traceability**
Machine Learning Pipeline

Models Only Give **Probabilistic Guarantees**

- data
- data preparation
- model training
- model deployment
- predictions

Stop + Maximum Speed 100 = STOP

*not sufficient for guaranteeing an acceptable failure rate under any circumstance*
Formal Methods
Mathematical Guarantees of Safety

Deductive Verification
- extremely expressive
- relies on the user to guide the proof

Model Checking
- analysis of a model of the software
- sound and complete with respect to the model

Static Analysis
- analysis of the software at some level of abstraction
- fully automatic and sound by construction
- generally not complete
Formal Methods for Trained Models
Neural Networks
Neural Networks

Feed-Forward Fully-Connected Neural Networks with ReLU Activation Functions

\[
x_{i,j} = \max \left\{ \sum_k w_{j,k} \cdot x_{i-1,k} + b_{i,j}, 0 \right\}
\]

Rectified Linear Unit (ReLU)

input layer

hidden layers

output layer
Feed-Forward Fully-Connected ReLU Networks as Programs

\[
x_{00} = \text{input}() \\
x_{01} = \text{input}() \\
x_{10} = -0.31 * x_{00} + 0.99 * x_{01} + (-0.63) \\
x_{11} = -1.25 * x_{00} + (-0.64) * x_{01} + 1.88 \\
x_{20} = 0 \text{ if } x_{10} < 0 \text{ else } x_{10} \\
x_{21} = 0 \text{ if } x_{11} < 0 \text{ else } x_{11} \\
x_{20} = 0.40 * x_{10} + 1.21 * x_{11} + 0.00 \\
x_{21} = 0.64 * x_{10} + 0.69 * x_{11} + (-0.39) \\
x_{30} = 0.26 * x_{20} + 0.33 * x_{21} + 0.45 \\
x_{31} = 1.42 * x_{20} + 0.40 * x_{21} + (-0.45) \\
return \ ' ' \text{ if } x_{31} < 30 \text{ else } \ ' ' \\
\]
Maximal Trace Semantics

\[ M \]

\begin{align*}
x_{00} &= \text{input}() \\
x_{01} &= \text{input}() \\
x_{10} &= -0.31 \cdot x_{00} + 0.99 \cdot x_{01} + (-0.63) \\
x_{11} &= -1.25 \cdot x_{00} + (-0.64) \cdot x_{01} + 1.88 \\
x_{10} &= 0 \text{ if } x_{10} < 0 \text{ else } x_{10} \\
x_{11} &= 0 \text{ if } x_{11} < 0 \text{ else } x_{11} \\
x_{20} &= 0.40 \cdot x_{10} + 1.21 \cdot x_{11} + 0.00 \\
x_{21} &= 0.64 \cdot x_{10} + 0.69 \cdot x_{11} + (-0.39) \\
x_{20} &= 0 \text{ if } x_{20} < 0 \text{ else } x_{20} \\
x_{21} &= 0 \text{ if } x_{21} < 0 \text{ else } x_{21} \\
x_{30} &= 0.26 \cdot x_{20} + 0.33 \cdot x_{21} + 0.45 \\
x_{31} &= 1.42 \cdot x_{20} + 0.40 \cdot x_{21} + (-0.45) \\
\text{return } '\text{green}' \text{ if } x_{31} < 30 \text{ else } '\text{red}'
\end{align*}
Neural Network Verification
Collecting Semantics

The collecting semantics \( \text{Col} : \text{Prog} \rightarrow \mathcal{P}(\mathcal{P}(\Sigma^*)) \) is the strongest property of a program.

Hence: \( \text{Col}(\text{prog}) \stackrel{\text{def}}{=} \{ \llbracket \text{prog} \rrbracket \} \)

**Benefit:**
- given a program \( \text{prog} \) and a property \( P \in \mathcal{P}(\mathcal{P}(\Sigma^*)) \) the verification problem is an inclusion checking:
  \( \text{Col}(\text{prog}) \subseteq P \)
- generally, the collecting semantics cannot be computed, we settle for a weaker property \( S^\sharp \) that
  - is sound: \( \text{Col}(\text{prog}) \subseteq S^\sharp \)
  - implies the desired property: \( S^\sharp \subseteq P \)

\[
\begin{align*}
x_{00} &= \text{input}() \\
x_{01} &= \text{input}() \\
x_{10} &= -0.31 \times x_{00} + 0.99 \times x_{01} + (-0.63) \\
x_{11} &= -1.25 \times x_{00} + (-0.64) \times x_{01} + 1.88 \\
x_{20} &= 0 \text{ if } x_{10} < 0 \text{ else } x_{10} \\
x_{21} &= 0 \text{ if } x_{11} < 0 \text{ else } x_{11} \\
x_{30} &= 0.40 \times x_{20} + 1.21 \times x_{11} + 0.00 \\
x_{31} &= 0.64 \times x_{20} + 0.69 \times x_{11} + (-0.39) \\
x_{20} &= 0.26 \times x_{20} + 0.33 \times x_{21} + 0.45 \\
x_{31} &= 1.42 \times x_{20} + 0.40 \times x_{21} + (-0.45) \\
\text{return '         ' if } x_{31} < 30 \text{ else '         '}
\end{align*}
\]
Stability
Goal G3 in [Kurd03]

Safety
Goal G4 in [Kurd03]

Fairness
Stability
Goal G3 in [Kurd03]

Safety
Goal G4 in [Kurd03]

Fairness
Local Stability

The classification is unaffected by small input perturbations.
Local Stability

Distance-Based Perturbations

\[ P_{\delta, \epsilon}(x) \overset{\text{def}}{=} \{ x' \in \mathcal{R}^{L_0} \mid \delta(x, x') \leq \epsilon \} \]

Example (\(L_\infty\) distance): \( P_{\infty, \epsilon}(x) \overset{\text{def}}{=} \{ x' \in \mathcal{R}^{L_0} \mid \max_i |x_i - x'_i| \leq \epsilon \} \)

\[ \mathcal{R}_{x}^{\delta, \epsilon} \overset{\text{def}}{=} \{ \llbracket M \rrbracket \in \mathcal{P}(\Sigma^*) \mid \text{STABLE}^{\delta, \epsilon}_{x}(\llbracket M \rrbracket) \} \]

\( \mathcal{R}_{x}^{\delta, \epsilon} \) is the set of all neural networks \( M \) (or, rather, their semantics \( \llbracket M \rrbracket \)) that are stable in the neighborhood \( P_{\delta, \epsilon}(x) \) of a given input \( x \)

\[ \text{STABLE}^{\delta, \epsilon}_{x}(\llbracket M \rrbracket) \overset{\text{def}}{=} \forall t \in \llbracket M \rrbracket : (\exists t' \in \llbracket M \rrbracket : \forall 0 \leq i \leq |L_0| : t'_0(x_{0,i}) = x_i) \]
\[ \land (\exists x' \in P_{\delta, \epsilon}(x) : \forall 0 \leq i \leq |L_0| : t_0(x_{0,i}) = x'_i) \]
\[ \Rightarrow \max_j t_\omega(x_{N,j}) = \max_j t'_\omega(x_{N,j}) \]

**Theorem**

\[ M \models \mathcal{R}_{x}^{\delta, \epsilon} \iff \{ \llbracket M \rrbracket \} \subseteq \mathcal{R}_{x}^{\delta, \epsilon} \]

**Corollary**

\[ M \models \mathcal{R}_{x}^{\delta, \epsilon} \iff \llbracket M \rrbracket \subseteq \bigcup \mathcal{R}_{x}^{\delta, \epsilon} \]
Static Analysis Methods
Forward Analysis

1. Proceed *forwards from an abstraction* of all possible perturbations

2. Check output for *inclusion* in expected output:
   - Included → *stable*
   - Otherwise → *alarm*
Example

\[ P(\langle 0.5, 0.75 \rangle) \overset{\text{def}}{=} \{ \mathbf{x} \in \mathbb{R} \times \mathbb{R} \mid 0 \leq x_0 \leq 1 \land 0 \leq x_1 \leq 1 \} \]
Interval Domain

\[ x_{i,j} \mapsto [a, b] \quad a, b \in \mathbb{R} \]

\[ x_{00} \mapsto [0, 1] \]

\[ x_{10} \mapsto [4, 6] \]

ReLU

\[ x_{10} \mapsto [4, 6] \]

\[ x_{20} \mapsto [17, 24] \]

ReLU

\[ x_{20} \mapsto [17, 24] \]

\[ x_{30} \mapsto [0, 10] \]

not precise enough!

\[ x_{01} \mapsto [0, 1] \]

\[ x_{11} \mapsto [3, 4] \]

ReLU

\[ x_{11} \mapsto [3, 4] \]

\[ x_{21} \mapsto [0, 3] \]

ReLU

\[ x_{21} \mapsto [0, 3] \]

\[ x_{31} \mapsto [-4, 4] \]
Interval Domain
with Symbolic Constant Propagation \[\text{[Li19]}\]

\[x_{i,j} \mapsto \begin{cases} 
\sum_{k=0}^{i-1} c_k \cdot x_k + c & c_k, c \in \mathcal{R}|X_k| \\
[a, b] & a, b \in \mathcal{R}
\end{cases}\]

\[x_{i-1,0} \mapsto E_{i-1,0}\]
\[\ldots\]
\[x_{i-1,j} \mapsto E_{i-1,j}\]
\[\ldots\]

\[x_{i,j} = \sum_k w_{i,k} \cdot x_{i-1,k} + b_{i,j}\]

\[x_{i,j} \mapsto \sum_k w_{i,k} \cdot E_{i-1,k} + b_{i,j}\]

\[x_{i,j} \mapsto \begin{cases} 
E_{i,j} & \text{[a, b]} \\
0 & a \leq 0 \wedge 0 < b
\end{cases}\]

\[x_{i,j} \mapsto \begin{cases} 
x_{i,j} & \text{[0, b]} \\
0 & b \leq 0
\end{cases}\]

\[x_{i,j} \mapsto \begin{cases} 
0 & [0, 0]
\end{cases}\]

---

J. Li et al. - Analyzing Deep Neural Networks with Symbolic Propagation (SAS 2019)
Interval Domain
with Symbolic Constant Propagation [Li19]

\[ x_{10} \mapsto \begin{cases} x_{00} + x_{01} + 4 \\ [4, 6] \end{cases} \]

\[ x_{20} \mapsto \begin{cases} 2 \cdot (x_{00} + x_{01} + 4) + 3 \cdot (0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3) \\ [17, 24] \end{cases} \]

\[ x_{30} \mapsto \begin{cases} 3 \cdot x_{00} + 3 \cdot x_{01} + 2 \\ [2, 8] \end{cases} \]

\[ x_{31} \mapsto \begin{cases} x_{00} + x_{01} - 1 \\ [-1, 1] \end{cases} \]
Interval Domain
with Symbolic Constant Propagation [Li19]

\[ x_{40} \mapsto \begin{cases} 1.5 \cdot x_{00} + 1.5 \cdot x_{01} + 2 \cdot x_{31} + 2 \\ [0, 5] \end{cases} \]

\[ x_{30} \mapsto \begin{cases} 3 \cdot x_{00} + 3 \cdot x_{01} + 2 \\ [2, 8] \end{cases} \]

\[ x_{31} \mapsto \begin{cases} x_{00} + x_{01} - 1 \\ [-1, 1] \end{cases} \]

\[ x_{41} \mapsto \begin{cases} x_{31} \\ [0, 1] \end{cases} \]

ReLU

not precise enough!
DeepPoly \textsuperscript{[Singh19]}

\[ x_{i+1,j} \mapsto \begin{cases} 
[\sum_k c_{i,k} \cdot x_{i,k} + c, \sum_k d_{i,k} \cdot x_{i,k} + d] & c_{i,k}, c, d_{i,k}, d \in \mathcal{R} \\
[a, b] & a, b \in \mathcal{R}
\end{cases} \]

\[ x_{i,j} \mapsto \begin{cases} 
[0, \frac{b(x_{i,j} - a)}{b - a}] & b \leq -a \\
[0, b] & x_{i,j} \leq \frac{b(x_{i,j} - a)}{b - a} \\
[a, b] & b \leq 0 \\
[a, b] & a < 0 \land 0 < b \\
\{0, 0\} & -a < b \]

G. Singh, T. Gehr, M. Püschel, and M. Vechev - An Abstract Domain for Certifying Neural Networks (POPL 2019)
DeepPoly [Singh19]

\[ x_{10} \mapsto \{ [x_{00} + x_{01} + 4, x_{00} + x_{01} + 4] \} \]

\[ x_{00} \mapsto \{ [x_{00}, x_{00}] \} \]

\[ x_{01} \mapsto \{ [x_{01}, x_{01}] \} \]

\[ x_{11} \mapsto \{ [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3, 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3] \} \]

\[ [3, 4] \]
DeepPoly [Singh19]

\[
x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] & \text{0.5} \\ [0, 1] & \text{0.5} \end{cases}
\]

\[
x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] & \text{0.5} \\ [0, 1] & \text{0.5} \end{cases}
\]

\[
x_{20} \mapsto \begin{cases} [2 \cdot x_{10} + 3 \cdot x_{11}, 2 \cdot x_{10} + 3 \cdot x_{11}] & \text{0.5} \\ [17, 24] & \text{0.5} \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} [x_{10} - x_{11}, x_{10} - x_{11}] & \text{0.5} \\ [1, 2] & \text{0.5} \end{cases}
\]
DeepPoly [Singh19]

\[
x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] & 0.5 \\ [0, 1] & \end{cases}
\]

\[
x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] & 0.5 \\ [0, 1] & \end{cases}
\]

ReLU

\[
ReLU(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \\ \end{cases}
\]

\[
x_{30} \mapsto \begin{cases} [x_{20} - x_{21} - 14, x_{20} - x_{21} - 14] & \end{cases}
\]

\[
x_{31} \mapsto \begin{cases} [0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8, 0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8] & \\ [-1, 1] & \end{cases}
\]

\[
x_{31} \mapsto \begin{cases} [0, 0.5 \cdot x_{31} + 0.5] & \\ [0, 1] & \end{cases}
\]
DeepPoly [Singh19]

\[ x_{40} \mapsto \left\{ 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1, 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1 \right\} \]
DeepPoly [Singh19]

\[ x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] \\ [0, 1] \end{cases} \]
\[ x_{10} \mapsto \begin{cases} [x_{00} + x_{01} + 4, x_{00} + x_{01} + 4] \\ [4, 6] \end{cases} \]
\[ x_{20} \mapsto \begin{cases} [2 \cdot x_{10} + 3 \cdot x_{11}, 2 \cdot x_{10} + 3 \cdot x_{11}] \\ [17, 24] \end{cases} \]
\[ x_{30} \mapsto \begin{cases} [x_{20} - x_{21} - 14, x_{20} - x_{21} - 14] \\ [2, 8] \end{cases} \]
\[ x_{40} \mapsto \begin{cases} [0.5 \cdot x_{30} - 2 \cdot x_{31} + 1, 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1] \\ \quad \mapsto \begin{cases} [x_{21} + 1, 0.5 \cdot x_{20} - 0.5 \cdot x_{21} - 6] \\ \quad \quad \mapsto \begin{cases} [x_{10} - x_{11} + 1, 0.5 \cdot x_{10} + 2 \cdot x_{11} - 6] \\ \quad \quad \quad \mapsto \begin{cases} [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 2, 1.5 \cdot x_{00} + 1.5 \cdot x_{11} + 2] \\ [2, 5] \end{cases} \end{cases} \end{cases} \]

\[ x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] \\ [0, 1] \end{cases} \]
\[ x_{11} \mapsto \begin{cases} [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3, 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3] \\ [3, 4] \end{cases} \]
\[ x_{21} \mapsto \begin{cases} [x_{10} - x_{11}, x_{10} - x_{11}] \\ [1, 2] \end{cases} \]
\[ x_{31} \mapsto \begin{cases} [0, 0.5 \cdot (0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8) + 0.5] \\ [0, 1] \end{cases} \]
DeepPoly \cite{Singh19}

$x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] \\ [0, 1] \end{cases}$

$x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] \\ [0, 1] \end{cases}$

$x_{40} \mapsto \begin{cases} 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1, \\ 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1 \end{cases} \in [2, 5]$

$x_{41} \mapsto \begin{cases} [x_{31}, x_{31}] \end{cases}$
DeepPoly [Singh19]

\[
x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] \\ [0, 1] \end{cases}
\]

\[
x_{10} \mapsto \begin{cases} [x_{00} + x_{01} + 4, x_{00} + x_{01} + 4] \\ [4, 6] \end{cases}
\]

\[
x_{20} \mapsto \begin{cases} [2 \cdot x_{10} + 3 \cdot x_{11}, 2 \cdot x_{10} + 3 \cdot x_{11}] \\ [17, 24] \end{cases}
\]

\[
x_{30} \mapsto \begin{cases} [x_{20} - x_{21} - 14, x_{20} - x_{21} - 14] \\ [2, 8] \end{cases}
\]

\[
x_{40} \mapsto \begin{cases} [x_{31}, x_{31}] \\ \end{cases}
\]

\[
x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] \\ [0, 1] \end{cases}
\]

\[
x_{11} \mapsto \begin{cases} [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3, 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3] \\ [3, 4] \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} [x_{10} - x_{11}, x_{10} - x_{11}] \\ [1, 2] \end{cases}
\]

\[
x_{31} \mapsto \begin{cases} [0, 0.5 \cdot (0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8) + 0.5] \\ [0, 1] \end{cases}
\]

\[
x_{41} \mapsto \begin{cases} [0, 0.25 \cdot x_{20} - 0.75 \cdot x_{21} - 3.5] \\ \end{cases}
\]

\[
x_{41} \mapsto \begin{cases} [0, -0.25 \cdot x_{10} + 1.5 \cdot x_{11} - 3.5] \\ \end{cases}
\]

\[
x_{41} \mapsto \begin{cases} [0, 0.5 \cdot x_{00} + 0.5 \cdot x_{01}] \\ [0, 1] \end{cases}
\]
DeepPoly \cite{Singh19}

\[ x_{40} \mapsto \begin{cases} [0.5 \cdot x_{30} - 2 \cdot x_{31} + 1, 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1] \\ [2, 5] \end{cases} \]

\[ x_{41} \mapsto \begin{cases} [x_{31}, x_{31}] \\ [0, 1] \end{cases} \]
Other Static Analysis Methods

  the first use of abstract interpretation for verifying neural networks

  a custom zonotope domain for certifying neural networks

  a framework to jointly approximate k ReLU activations

  a multi-neuron abstraction via a convex-hull approximation algorithm
Stability
Goal G3 in [Kurd03]

Safety
Goal G4 in [Kurd03]

Fairness

A nurse
A doctor

Google Translate

A nurse
Une infirmière

A doctor
Un médecin
ACAS Xu [Julian16][Katz17]

Airborne Collision Avoidance System for Unmanned Aircraft
implemented using 45 feed-forward fully-connected ReLU networks

5 input sensor measurements
- $\rho$: distance from ownship to intruder
- $\theta$: angle to intruder relative to ownship heading direction
- $\psi$: heading angle to intruder relative to ownship heading direction
- $v_{own}$: speed of ownship
- $v_{int}$: speed of intruder

5 output horizontal advisories
- Strong Left
- Weak Left
- Clear of Conflict
- Weak Right
- Strong Right
ACAS Xu Properties [Katz17]

Example: “if intruder is near and approaching from the left, go Strong Right”

\[250 \leq \rho \leq 400\]

\[0.2 \leq \theta \leq 0.4\]

\[\ldots\]

\[\ldots\]

\[\ldots\]

\[\ldots\]
Safety

Input-Output Properties

\( \mathbf{I} \): input specification

\( \mathbf{O} \): output specification

\[ \mathcal{S}_{\mathbf{I}}^{\mathbf{O}} \overset{\text{def}}{=} \{ [[M]] \in \mathcal{P}(\Sigma^*) \mid \text{SAFE}_{\mathbf{O}}^{\mathbf{I}}([[M]]) \} \]

\( \mathcal{S}_{\mathbf{I}}^{\mathbf{O}} \) is the set of all neural networks \( M \) (or, rather, their semantics \( [[M]] \)) that satisfy the input and output specification \( \mathbf{I} \) and \( \mathbf{O} \)

\[ \text{SAFE}_{\mathbf{O}}^{\mathbf{I}}([[M]]) \overset{\text{def}}{=} \forall t \in [[M]] : t_0 \vDash I \Rightarrow t_\omega \vDash O \]

Theorem

\[ M \vDash \mathcal{S}_{\mathbf{I}}^{\mathbf{O}} \iff \{ [[M]] \} \subseteq \mathcal{S}_{\mathbf{I}}^{\mathbf{O}} \]

Corollary

\[ M \vDash \mathcal{S}_{\mathbf{I}}^{\mathbf{O}} \iff [[M]] \subseteq \bigcup \mathcal{S}_{\mathbf{I}}^{\mathbf{O}} \]
Model Checking Methods
Safety

Example

\[ l_j \leq x_{0,j} \leq u_j \]

\[ x_N > 0 \]
SMT-Based Methods

Verification Reduced to Constraint Satisfiability

\[ l_j \leq x_{0,j} \leq u_j \quad j \in \{0, \ldots, |X_0|\} \]

\[ \hat{x}_{i+1,j} = \sum_{k=0}^{\frac{|X_j|}{2}} w_{j,k} \cdot x_{i,k} + b_{i,j} \quad i \in \{0, \ldots, n - 1\} \]

\[ x_{i,j} = \max\{0, \hat{x}_{i,j}\} \quad i \in \{1, \ldots, n - 1\}, \quad j \in \{0, \ldots, |X_i|\} \]

\[ x_N \leq 0 \]

input specification

\( \neg \) input specification

output specification

satisfiable \( \rightarrow \) counterexample

otherwise \( \rightarrow \) safe
Planet

\[ x_{i,j} = \max\{0, \hat{x}_{i,j}\} \]

use approximations to reduce the solution search space

\[ 0 \leq x_{i,j} \]
\[ \hat{x}_{i,j} \leq x_{i,j} \]
\[ x_{i,j} \leq \frac{b_{i,j}}{b_{i,j} - a_{i,j}} \cdot (\hat{x}_{i,j} - a_{i,j}) \]

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

based on the simplex algorithm extended to support ReLUs

G. Katz et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)
Reluplex

G. Katz et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)

Follow-up Work
G. Katz et al. - The Marabou Framework for Verification and Analysis of Deep Neural Networks (CAV 2019)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{00}$</td>
<td>$v_{00}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\hat{x}_{ij}$</td>
<td>$\hat{v}_{ij}$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>$v_{ij}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_N$</td>
<td>$v_N$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{x}_{ij}$</td>
<td>$\hat{v}_{ij}'$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>$v_{ij}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_N$</td>
<td>$v_N$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{00}$</td>
<td>$v_{00}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\hat{x}_{ij}$</td>
<td>$\hat{v}_{ij}'$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>$v_{ij}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_N$</td>
<td>$v_N$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{00}$</td>
<td>$v_{00}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\hat{x}_{ij}$</td>
<td>$\hat{v}_{ij}'$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>$v_{ij}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x_N$</td>
<td>$v_N$</td>
</tr>
</tbody>
</table>

G. Katz et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)
Other SMT-Based Methods

• **L. Pulina and A. Tacchella.** *An Abstraction-Refinement Approach to Verification of Artificial Neural Networks.* In CAV, 2010. 
  the first formal verification method for neural networks

  an approach for finding the nearest adversarial example according to the \( L_{\infty} \) distance

• **X. Huang, M. Kwiatkowska, S. Wang, and M. Wu.** *Safety Verification of Deep Neural Networks.* In CAV, 2017. 
  an approach for proving local robustness to adversarial perturbations

  approaches focusing on binarized neural networks
MILP-Based Methods

Verification Reduced to Mixed Integer Linear Program

\[ l_j \leq x_{0,j} \leq u_j \quad j \in \{0, \ldots, |\mathbf{X}_0|\} \]

\[ \hat{x}_{i+1,j} = \sum_{k=0}^{\frac{|\mathbf{X}_i|}{\mathbf{X}_i}} w_{j,k}^i \cdot x_{i,k} + b_{i,j} \quad i \in \{0, \ldots, n - 1\} \]

\[ x_{i,j} = \delta_{i,j} \cdot \hat{x}_{i,j} \quad \delta_{i,j} \in \{0, 1\} \]

\[ \delta_{i,j} = 1 \Rightarrow \hat{x}_{i,j} \geq 0 \quad i \in \{1, \ldots, n - 1\} \]

\[ \delta_{i,j} = 0 \Rightarrow \hat{x}_{i,j} < 0 \quad j \in \{0, \ldots, |\mathbf{X}_i|\} \]

\[ \min \mathbf{X}_N \]

\[ \min \mathbf{X}_N \leq 0 \rightarrow \text{counterexample} \]

otherwise \[ \rightarrow \text{safe} \]
MILP-Based Methods

Bounded Encoding with Symmetric Bounds

\[ \hat{x}_{i+1,j} = \sum_{k=0}^{\left| X_i \right|} w_{j,k}^i \cdot x_{i,k} + b_{i,j} \]

\[ 0 \leq x_{i,j} \leq M_{i,j} \cdot \delta_{i,j} \]

\[ \hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - M_{i,j} \cdot (1 - \delta_{i,j}) \]

\[ M_{i,j} = \max\{-l_i, u_i\} \]

\[ i \in \{0,\ldots,n - 1\} \]

\[ \delta_{i,j} \in \{0, 1\} \]

\[ i \in \{1,\ldots,n - 1\} \]

\[ j \in \{0,\ldots,\left| X_i \right|\} \]
Sherlock

Output Range Analysis

\[ l_j \leq x_{0,j} \leq u_j \]

\[ \hat{x}_{i+1,j} = \sum_{k=0}^{\lfloor X_i \rfloor} w_{j,k} \cdot x_{i,k} + b_{i,j} \]

\[ 0 \leq x_{i,j} \leq M_{i,j} \cdot \delta_{i,j} \]

\[ \hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - M_{i,j} \cdot (1 - \delta_{i,j}) \]

\[ M_{i,j} = \max\{-l_i, u_i\} \]

\[ \min \ x_N \]

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)
### Sherlock

#### Output Range Analysis

\[ l_j \leq x_{0,j} \leq u_j \]

\[ \hat{x}_{i+1,j} = \sum_{k=0}^{\left| X_i \right|} w_{j,k} \cdot x_{i,k} + b_{i,j} \]

\[ 0 \leq x_{i,j} \leq M_{i,j} \cdot \delta_{i,j} \]

\[ \hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - M_{i,j} \cdot (1 - \delta_{i,j}) \]

\[ M_{i,j} = \max\{-l_{i,j}, u_{i,j}\} \]

\[ x_N < L \]

---

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)

---

use **local search** to speed up the MILP solver

**sample** random input \( X \)

and evaluate output \( L \)
Sherlock

Output Range Analysis

\[ l_j \leq x_{0,j} \leq u_j \]

\[ \hat{x}_{i+1,j} = \sum_{k=0}^{\lfloor \frac{|X|}{i} \rfloor} w_{j,k} \cdot x_{i,k} + b_{i,j} \]

\[ 0 \leq x_{i,j} \leq M_{i,j} \cdot \delta_{i,j} \]

\[ \hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - M_{i,j} \cdot (1 - \delta_{i,j}) \]

\[ M_{i,j} = \max\{-l_i, u_i\} \]

\[ x_N < L \]

---

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)
### Sherlock

**Output Range Analysis**

\[
\begin{align*}
l_j & \leq x_{0,j} \leq u_j \\
\hat{x}_{i+1,j} & = \sum_{k=0}^{\left|X_i\right|} w_{j,k}^i \cdot x_{i,k} + b_{i,j} \\
0 & \leq x_{i,j} \leq M_{i,j} \cdot \delta_{i,j} \\
\hat{x}_{i,j} & \leq x_{i,j} \leq \hat{x}_{i,j} - M_{i,j} \cdot (1 - \delta_{i,j}) \\
M_{i,j} & = \max\{-l_i, u_i\} \\
x_N < \hat{L}
\end{align*}
\]

*use local search to speed up the MILP solver*

*find another input \( \hat{X} \) such that \( \hat{L} \leq x_N \)*

---

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)
**MILP-Based Methods**

**Bounded Encoding with Asymmetric Bounds**

\[
\hat{x}_{i+1,j} = \sum_{k=0}^{\left|X_i\right|} w_{j,k}^i \cdot x_{i,k} + b_{i,j}, \quad i \in \{0,\ldots, n - 1\}
\]

\[
0 \leq x_{i,j} \leq u_{i,j} \cdot \delta_{i,j}, \quad \delta_{i,j} \in \{0, 1\}
\]

\[
\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - l_{i,j} \cdot (1 - \delta_{i,j}), \quad i \in \{1,\ldots, n - 1\}, \quad j \in \{0,\ldots, \left|X_i\right|\}
\]
MIPVerify

Finding Nearest Adversarial Example

\[
\min_{x'} d(X, X')
\]

\[
\hat{x}_{i+1,j} = \sum_{k=0}^{\left| X_i \right|} w_{j,k}^i \cdot x_{i,k} + b_{i,j}
\]

\[
i \in \{0, \ldots, n - 1\}
\]

\[
0 \leq x_{i,j} \leq u_{i,j} \cdot \delta_{i,j}
\]

\[
\delta_{i,j} \in \{0, 1\}
\]

\[
\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - l_{i,j} \cdot (1 - \delta_{i,j})
\]

\[
i \in \{1, \ldots, n - 1\}
\]

\[
j \in \{0, \ldots, |X_i|\}
\]

\[x_N \neq 0\]

V. Tjeng et al. - Evaluating Robustness of Neural Networks with Mixed Integer Programming (ICLR 2019)
Other MILP-Based Methods


Static Analysis Methods
Forward Analysis

1. Proceed forwards from an abstraction of the input specification \( I \)

2. Check output for inclusion in output specification \( O \):
   - Included → safe
   - Otherwise → alarm

\( I \)
Example

\[ 0 \leq \rho \leq 1 \]

\[ -1 \leq \theta \leq 1 \]

Clear of Conflict
DeepPoly Domain [Singh19]

$x_{10} \mapsto \begin{cases} [x_{00}, \frac{2}{3} \cdot x_{10} + \frac{2}{3}] & \text{if } -1 \leq x_{10} \\ [-1, 2] & \text{otherwise} \end{cases}$

$x_{10} \mapsto \begin{cases} [x_{00} + x_{01}, x_{00} + x_{01}] & \text{if } -1 \leq x_{10} \\ [-1, 2] & \text{otherwise} \end{cases}$

$x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] & \text{if } 0 \leq x_{00} \\ [0, 1] & \text{otherwise} \end{cases}$

$x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] & \text{if } -1 \leq x_{01} \\ [-1, 1] & \text{otherwise} \end{cases}$

$0 \leq \rho \leq 1$

$-1 \leq \theta \leq 1$

ReLU

Clear of Conflict

Strong Turn
DeepPoly Domain [Singh19]

\[ x_{00} \mapsto \left\{ \begin{array}{c} [x_{00}, x_{00}] \\ [0, 1] \end{array} \right\} \]

\[ x_{01} \mapsto \left\{ \begin{array}{c} [x_{01}, x_{01}] \\ [-1, 1] \end{array} \right\} \]

\[ -1 \leq \theta \leq 1 \]

\[ 0 \leq \rho \leq 1 \]

\[ x_{20} \mapsto \left\{ \begin{array}{c} [x_{10} + x_{11}, x_{10} + x_{11}] \\ [0, \frac{8}{3}] \end{array} \right\} \]

Clear of Conflict

Strong Turn

ReLU

\[ \text{ReLU}(x) \]

\[ \text{ReLU}(x) \leq \frac{b(x - a)}{b - a} \]

\[ a \leq \text{ReLU}(x) \leq b \]

\[ x_{21} \mapsto \left\{ \begin{array}{c} [x_{10} - x_{11}, x_{10} - x_{11}] \\ [\frac{-7}{3}, \frac{7}{3}] \end{array} \right\} \]

\[ x_{21} \mapsto \left\{ \begin{array}{c} [0, 0.5 \cdot x_{21} + \frac{7}{6}] \\ [0, \frac{7}{3}] \end{array} \right\} \]
DeepPoly Domain [Singh19]

\[
x_{30} \mapsto \begin{cases} 
{x_{20} + x_{21} + 1, x_{20} + x_{21} + 1} \\
[1, 5.5]
\end{cases}
\]

\[
x_{31} \mapsto \begin{cases} 
{x_{21} - 1.25, x_{21} - 1.25} \\
[-1.25, \frac{13}{12}]
\end{cases}
\]

\[
x_{00} \mapsto \begin{cases} 
[x_{00}, x_{00}] \\
[0, 1]
\end{cases}
\]

\[
x_{01} \mapsto \begin{cases} 
[x_{01}, x_{01}] \\
[-1, 1]
\end{cases}
\]

\[
0 \leq \rho \leq 1
\]

\[
-1 \leq \theta \leq 1
\]

Clear of Conflict

not precise enough!

Strong Turn
Interval Domain
with Symbolic Constant Propagation [Li19]

\[ x_{00} \mapsto \begin{cases} x_{00} \\ [0, 1] \end{cases} \]

\[ x_{01} \mapsto \begin{cases} x_{01} \\ [-1, 1] \end{cases} \]

\[ -1 \leq \rho \leq 1 \]

\[ x_{10} \mapsto \begin{cases} x_{00} + x_{01} \\ [-1, 2] \end{cases} \]

\[ x_{10} \mapsto \begin{cases} x_{10} \\ [0, 2] \end{cases} \]

ReLU

Clear of Conflict

Strong Turn

ReLU
Interval Domain
with Symbolic Constant Propagation \[\text{[Li19]}\]

\[
x_00 \mapsto \begin{cases} x_00 \\ [0, 1] \end{cases}
\]

\[
x_{01} \mapsto \begin{cases} x_{01} \\ [-1, 1] \end{cases}
\]

\[
0 \leq \rho \leq 1
\]

\[
-1 \leq \theta \leq 1
\]

\[
x_{20} \mapsto \begin{cases} x_{10} + x_{11} \\ [0, 4] \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} x_{10} - x_{11} \\ [-2, 2] \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} x_{21} \\ [0, 2] \end{cases}
\]

Clear of Conflict

Strong Turn
**Interval Domain**

with **Symbolic Constant Propagation** [Li19]

**DeepPoly Domain** [Singh19]

```
x_{30} \rightarrow \begin{cases} 
  x_{10} + x_{11} + x_{21} + 1 \\
  [1, 7]
\end{cases}
```

Clear of Conflict

```
x_{30} \rightarrow \begin{cases} 
  x_{20} + x_{21} + 1 \\
  [1, 7]
\end{cases}
```

Clear of Conflict

```
x_{31} \rightarrow \begin{cases} 
  x_{21} - 1.25 \\
  [-1.25, 0.75]
\end{cases}
```

not precise enough!
Product Domain [Mazzucato21]

DeepPoly with Symbolic Constant Propagation

\[ x_{00} \rightarrow \begin{cases} x_{00} \\ [x_{00}, x_{00}] \\ [0, 1] \end{cases} \]

\[ x_{01} \rightarrow \begin{cases} x_{01} \\ [x_{01}, x_{01}] \\ [-1, 1] \end{cases} \]

\[ 0 \leq \rho \leq 1 \]

\[ -1 \leq \theta \leq 1 \]

Clear of Conflict

Strong Turn
Product Domain [Mazzucato21]

\[ x_{10} \mapsto \begin{cases} x_{10} + x_{11} & \rightarrow [0, 2] \\ [x_{10}, \frac{2}{3} \cdot x_{10} + \frac{2}{3}] & \rightarrow [-1, 2] \end{cases} \]

\[ x_{10} \mapsto \begin{cases} x_{00} + x_{01} & \rightarrow [0, 2] \\ [x_{00} + x_{01}, x_{00} + x_{01}] & \rightarrow [-1, 2] \end{cases} \]

\[ x_{00} \mapsto \begin{cases} x_{00} & \rightarrow [0, 1] \\ [x_{00}, x_{00}] & \rightarrow [0, 2] \end{cases} \]

\[ x_{01} \mapsto \begin{cases} x_{01} & \rightarrow [-1, 1] \\ [x_{01}, x_{01}] & \rightarrow [0, 1] \end{cases} \]

\[ x_{11} \mapsto \begin{cases} x_{11} & \rightarrow [0, 2] \\ [x_{11}, \frac{2}{3} \cdot x_{11} + \frac{2}{3}] & \rightarrow [-1, 2] \end{cases} \]

\[ x_{11} \mapsto \begin{cases} x_{00} - x_{01} & \rightarrow [-1, 2] \\ [x_{00} - x_{01}, x_{00} - x_{01}] & \rightarrow [0, 2] \end{cases} \]

Clear of Conflict

Strong Turn
Product Domain [Mazzucato21]

\[
x_{00} \mapsto \begin{cases} x_{00} \\ [x_{00}, x_{00}] \\ [0, 1] \end{cases}
\]

\[
x_{01} \mapsto \begin{cases} x_{01} \\ [x_{01}, x_{01}] \\ [-1, 1] \end{cases}
\]

\[
x_{20} \mapsto \begin{cases} x_{10} + x_{11} \\ [x_{10} + x_{11}, x_{10} + x_{11}] \\ [0, \frac{8}{3}] \end{cases}
\]

\[
x_{20} \mapsto \begin{cases} x_{10} + x_{11} \\ [x_{10} + x_{11}, x_{10} + x_{11}] \\ [0, \frac{8}{3}] \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} x_{21} \\ [0, 0.5 \cdot x_{21} + 0.5] \\ [0, \frac{5}{3}] \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} x_{21} \\ [0, 0.5 \cdot x_{21} + 0.5] \\ [0, \frac{5}{3}] \end{cases}
\]

\[
x_{21} \mapsto \begin{cases} x_{21} \\ [0, 0.5 \cdot x_{21} + 0.5] \\ [0, \frac{5}{3}] \end{cases}
\]
Product Domain \[\text{[Mazzucato21]}\]

\[x_{30} \mapsto \begin{cases} 
  x_{10} + x_{11} + x_{21} + 1 & \rightarrow [1, \frac{20}{3}] \\
  [x_{20} + x_{21} + 1, x_{20} + x_{21} + 1] & \rightarrow [1, 4.5] \\
  [1, 4.5] 
\end{cases} \]

\[x_{31} \mapsto \begin{cases} 
  x_{21} - 1.25 & \rightarrow [-1.25, \frac{5}{12}] \\
  [x_{21} - 1.25, x_{21} - 1.25] & \rightarrow [-1.25, \frac{5}{12}] \\
  [-1.25, \frac{5}{12}] 
\end{cases} \]
Other Complete Methods
Star Sets

Exact Static Analysis Method

\[ \Theta \overset{\text{def}}{=} \langle c, V, P \rangle \]

- \( c \in \mathbb{R}^n \): center
- \( V = \{v_1, \ldots, v_m\} \): basis vectors in \( \mathbb{R}^n \)
- \( P: \mathbb{R}^m \to \{\bot, T\} \): predicate

\[ [\Theta] = \{ x \mid x = c + \sum_{i=1}^{m} \alpha_i v_i \text{ such that } P(\alpha_1, \ldots, \alpha_m) = T \} \]

- fast and cheap affine mapping operations \( \rightarrow \) neural network layers
- inexpensive intersections with half-spaces \( \rightarrow \) ReLU activations

---

H.-D. Tran et al. - Star-Based Reachability Analysis of Deep Neural Networks (FM 2018)
Star Sets

Exact Static Analysis Method

\[ \Theta \overset{\text{def}}{=} \langle c, V, P \rangle \]

- \( c \in \mathbb{R}^n \): center
- \( V = \{ v_1, \ldots, v_m \} \): basis vectors in \( \mathbb{R}^n \)
- \( P: \mathbb{R}^m \to \{ \bot, T \} \): predicate

\[ \llbracket \Theta \rrbracket = \{ x \mid x = c + \sum_{i=1}^{m} \alpha_i v_i \text{ such that } P(\alpha_1, \ldots, \alpha_m) = T \} \]

- fast and cheap \text{affine mapping operations} \rightarrow \text{neural network layers}
- inexpensive \text{intersections with half-spaces} \rightarrow \text{ReLU activations}

---

H.-D. Tran et al. - Star-Based Reachability Analysis of Deep Neural Networks (FM 2018)
ReluVal

Asymptotically Complete Method

S. Wang et al. - Formal Security Analysis of Neural Networks Using Symbolic Intervals (USENIX Security 2018)
Neurify

Asymptotically Complete Method

\( x_{i,j} \mapsto \begin{cases} \left[ \sum_k c_{0,k} \cdot x_{0,k} + c, \sum_k d_{0,k} \cdot x_{0,k} + d \right] & \in \mathcal{R} \\ [a, b] \end{cases} \)

\( a, b \in \mathcal{R} \)

\( x_{i,j} \mapsto \begin{cases} \left[ E_{i,j}, E_{i,j} \right] & \in \mathcal{R} \\ [a, b] \end{cases} \)

\( x_{i,j} \mapsto \begin{cases} \left[ \frac{b}{b-a} E_{i,j}, \frac{b}{b-a} (E_{i,j} - a) \right] & \in \mathcal{R} \\ [a, b] \end{cases} \)

\( a < 0 \land 0 < b \)

\( b \leq 0 \)

\( 0 \leq a \)

\( \frac{a}{b-a} \leq \text{ReLU}(x) \leq \frac{b}{b-a} \)

S. Wang et al. - Formal Security Analysis of Neural Networks Using Symbolic Intervals (USENIX Security 2018)
Further Complete Methods


Other Incomplete Methods
Interval Neural Networks
Abstraction-Based Method

$l_j \leq x_{0,j} \leq u_j$

merge neurons layer-wise based on partitioning strategy + replace weights with intervals

Related Work
Y. Y. Elboher et al. - An Abstraction-Based Framework for Neural Network Verification (CAV 2020)

P. Prabhakar and Z. R. Afza - Abstraction based Output Range Analysis for Neural Networks (NeurIPS 2019)
Further Incomplete Methods


Further Incomplete Methods

- **A. Raghunathan, J. Steinhardt, and P. Liang.** *Certified Defenses against Adversarial Examples.* In ICML, 2018.

approaches for finding a lower bound on robustness to adversarial perturbations
Further Incomplete Methods

  approach focusing on convolutional neural networks

  approaches focusing on recurrent neural networks

  an approach for inferring safety properties of neural networks
Complete Methods

Advantages

sound and complete

Disadvantages

soundness not typically guaranteed with respect to floating-point arithmetic

do not scale to large models

often limited to certain model architectures

Disadvantages

suffer from false positives

able to scale to large models

sound often also with respect to floating-point arithmetic

less limited to certain model architectures

Advantages

Incomplete Methods
Stability
Goal G3 in [Kurd03]

Safety
Goal G4 in [Kurd03]

Fairness
ML Impacts Our Society

Machine Bias
There’s software used across the country to predict future criminals. And it’s biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica
May 23, 2016

Amazon scraps secret AI recruiting tool that showed bias against women

by Jeffrey Dastin

Can AI Be a Fair Judge in Criminal Cases? Estonia Thinks So

Estonia plans to use an artificial intelligence program to decide small-claims cases, part of a push to make government services smarter.
Translation tutorial:
21 fairness definitions and their politics

Arvind Narayanan
@random_walker

Tutorial: 21 fairness definitions and their politics
19,759 views · Mar 1, 2018

Arvind Narayanan
226 subscribers

Computer scientists and statisticians have devised numerous mathematical criteria to define what it means for a classifier or a model to be fair. The proliferation of these definitions represents an attempt to make technical sense of

SHOW MORE
Dependency Fairness [Galhotra17]

The classification is independent of the values of the sensitive inputs
Dependency Fairness

\[ \mathcal{F}_i \overset{\text{def}}{=} \{ [[M]] \in \mathcal{P}(\Sigma^*) | \text{UNUSED}_i([[M]]) \} \]

\( \mathcal{F}_i \) is the set of all neural networks \( M \) (or, rather, their semantics \([ [M] ]\)) that do not use the value of the sensitive input node \( x_{0,i} \) for classification.

\[ \text{UNUSED}_i([[M]]) \overset{\text{def}}{=} \forall t \in [[M]], v \in \mathcal{R} : t_0(x_{0,i}) \neq v \Rightarrow \exists t' \in [[M]] : \\
(\forall 0 \leq j \leq |L_0| : j \neq i \Rightarrow t_0(x_{0,j}) = t'_0(x_{0,j})) \\
\wedge t'_0(x_{0,i}) = v \\
\wedge \max_j t_\omega(x_{N,j}) = \max_j t'_\omega(x_{N,j}) \]

Intuitively: any possible classification outcome is possible from any value of the sensitive input node \( x_{0,i} \).

Input Data (Non-)Usage

\[ \mathcal{N}_J \overset{\text{def}}{=} \{ [[P]] \in \mathcal{P}(\Sigma^{+\infty}) | \forall i \in J \subseteq I_P : \text{UNUSED}_i([[P]]) \} \]

\( \mathcal{N}_J \) is the set of all programs \( P \) (or, rather, their semantics \([ [P] ]\)) that do not use the value of the input variables in \( J \subseteq I_P \).

\[ \text{UNUSED}_i([[M]]) \overset{\text{def}}{=} \forall t \in [[M]], v \in \mathcal{R} : t_0(i) \neq v \Rightarrow \exists t' \in [[M]] : \\
(\forall 0 \leq j \leq |I_P| : j \neq i \Rightarrow t_0(j) = t'_0(j)) \\
\wedge t'_0(i) = v \\
\wedge t'_\omega = t_\omega \]

Intuitively: any possible program outcome is possible from any value of the input variable \( i \).
Dependency Fairness

![Diagram showing dependency fairness]

- Initial situation with elderly people and money bags.
- Machine learning process.
- Final outcome with different results for elderly and younger people.

Dependency fairness is a concept in machine learning that aims to ensure that the predictions or decisions made by a model are not influenced by factors such as age, race, or gender. It is important to ensure that machine learning models are fair and unbiased, especially in critical applications like credit scoring or healthcare.
Dependency Fairness

\[ \mathcal{F}_i \overset{\text{def}}{=} \{ [[M]] \in \mathcal{P}(\Sigma^*) \mid \text{UNUSED}_i([[M]]) \} \]

\( \mathcal{F}_i \) is the set of all neural networks \( M \) (or, rather, their semantics \( [[M]] \)) that do not use the value of the sensitive input node \( x_{0,i} \) for classification.

\[ \text{UNUSED}_i([[M]]) \overset{\text{def}}{=} \forall t \in [[M]], v \in \mathcal{R} : t_0(x_{0,i}) \neq v \Rightarrow \exists t' \in [[M]] : \]
\[ (\forall 0 \leq j \leq |L_0| : j \neq i \Rightarrow t_0(x_{0,j}) = t'_0(x_{0,j})) \]
\[ \land t'_0(x_{0,i}) = v \]
\[ \land \max_j t_\omega(x_{N,j}) = \max_j t'_\omega(x_{N,j}) \]

Intuitively: any possible classification outcome is possible from any value of the sensitive input node \( x_{0,i} \).

Theorem

\[ M \models \mathcal{F}_i \iff \{ [[M]] \} \subseteq \mathcal{F}_i \]
Hierarchy of Semantics

parallel semantics

dependency semantics

outcome semantics

collecting semantics
Collecting Semantics

General collecting semantics
\[ \text{Col} : \text{Prog} \to \text{P}(\text{P}(\Sigma^*))) \]
Hence: \[ \text{Col}(\text{prog}) \subseteq \{\{\text{prog}\}\} \]

Benefits:
- Given a program \(\text{prog}\) and a property \(P \in \text{P}(\text{P}(\Sigma^*)))\) the verification problem is an inclusion checking:
  \[ \text{Col}(\text{prog}) \subseteq P \]
- Generally, the collecting semantics cannot be computed; we settle for a weaker property \(S^\#\) that
  - is sound: \(\text{Col}(\text{prog}) \subseteq S^\#\)
  - implies the desired property: \(S^\# \subseteq P\)

Outcome Semantics

\[ \{\{P\}\} \]

Partitioning a set of traces that satisfies input data (non-)usage with respect to the program outcome yields sets of traces that also satisfy input data (non-)usage.

Dependency Semantics

\[ \{P\} \]

To reason about input data (non-)usage, we do not need to consider all intermediate computations between the initial and final states of a trace (if any).
Dependency Semantics

Partitioning with respect to the outcome classification induces a partition of the space of values of the input nodes used for classification.

Lemma

\[ M \models \mathcal{F}_i \iff \forall A, B \in [M]_{\sim} : (A_\omega \neq B_\omega \Rightarrow A_0 \neq B_0 \neq i = \emptyset) \]
Naïve Abstraction
Naïve Backward Analysis

1. proceed backwards from all possible classification outcomes

2. **forget** the values of the **sensitive input** nodes

3. check for **intersection**: empty → **fair**
   otherwise → **alarm**
Naïve Backward Analysis

\[
x_{00} = \text{input}() \\
x_{01} = \text{input}() \\
x_{10} = -0.31 \times x_{00} + 0.99 \times x_{01} - 0.63 \\
x_{11} = -1.25 \times x_{00} - 0.64 \times x_{01} + 1.88 \\
x_{10} = 0 \text{ if } x_{10} < 0 \text{ else } x_{10} \\
x_{11} = 0 \text{ if } x_{11} < 0 \text{ else } x_{11} \\
x_{20} = 0.40 \times x_{10} + 1.21 \times x_{11} + 0.00 \\
x_{21} = 0.64 \times x_{10} + 0.69 \times x_{11} - 0.39 \\
x_{20} = 0 \text{ if } x_{20} < 0 \text{ else } x_{20} \\
x_{21} = 0 \text{ if } x_{21} < 0 \text{ else } x_{21} \\
x_{30} = 0.26 \times x_{20} + 0.33 \times x_{21} + 0.45 \\
x_{31} = 1.42 \times x_{20} + 0.40 \times x_{21} - 0.45 \\
x_{30} \geq x_{31} \text{ if } x_{31} < 30 \text{ else } \text{'} \\
\]

return \text{''} if } x_{31} < 30 \text{ else '}

\text{too many disjunctions!}
Back to the Semantics...
Hierarchy of Semantics

- **parallel semantics**
- **dependency semantics**
- **outcome semantics**
- **collecting semantics**
Parallel Semantics

partitioning a set of traces that satisfies dependency fairness with respect to the non-sensitive inputs yields sets of traces that also satisfy dependency fairness.
Parallel Semantics

\{[M]\}_{\sim}

Partitioning a set of traces that satisfies dependency fairness with respect to the non-sensitive inputs yields sets of traces that also satisfy dependency fairness.
Parallel Semantics

\[ \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)), \subseteq \rangle \rightarrow \langle \mathcal{P}(\mathcal{P}(\Sigma \times \Sigma)), \subseteq \rangle \]

\(\alpha_\parallel(S) \overset{\text{def}}{=} \{ \langle t_0, t_\omega \rangle \in R \mid t_0 \in I \} \mid R \in S \land I \in \bot\}

\{[[M]]_\parallel\} \overset{\text{def}}{=} \alpha_\parallel([[M]]_\parallel)

= \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma \mid t \in [[M]] \land t_0 \in I \land t_\omega \in O \} \mid I \in \bot \land O \in \emptyset\}

**Theorem**

\[ M \models \mathcal{F}_i \iff \gamma_\parallel([[M]]_\parallel) \subseteq \mathcal{F}_i \]

**Lemma**

\[ M \models \mathcal{F}_i \iff \forall I \in \bot : \forall A, B \in [[M]]_\parallel : (A_\omega^I \neq B_\omega^I \Rightarrow A_0^I|_{\neq i} \cap B_0^I|_{\neq i} = \emptyset) \]
Better Abstraction
Forward and Backward Analysis

1. **partition** the space of values of the **non-sensitive input** nodes

2. proceed **forwards** from all partitions to find:
   - already **fair** partitions
   - activation patterns

3. proceed **backwards** for each activation pattern
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
L = 0.25
U = 2

x00 = input()
x01 = input()
x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
\[
x_{00} = \text{input()}
\]
\[
x_{01} = \text{input()}
\]
\[
x_{10} = -0.31 \times x_{00} + 0.99 \times x_{01} + (-0.63)
\]
\[
x_{11} = -1.25 \times x_{00} + (-0.64) \times x_{01} + 1.88
\]
\[
x_{10} = 0 \quad \text{if} \quad x_{10} < 0 \quad \text{else} \quad x_{10}
\]
\[
x_{11} = 0 \quad \text{if} \quad x_{11} < 0 \quad \text{else} \quad x_{11}
\]
\[
x_{20} = 0.40 \times x_{10} + 1.21 \times x_{11} + 0.00
\]
\[
x_{21} = 0.64 \times x_{10} + 0.69 \times x_{11} + (-0.39)
\]
\[
x_{20} = 0 \quad \text{if} \quad x_{20} < 0 \quad \text{else} \quad x_{20}
\]
\[
x_{21} = 0 \quad \text{if} \quad x_{21} < 0 \quad \text{else} \quad x_{21}
\]
\[
x_{30} = 0.26 \times x_{20} + 0.33 \times x_{21} + 0.45
\]
\[
X_{31} = 1.42 \times x_{20} + 0.40 \times x_{21} + (-0.45)
\]
\[
\text{return 'thumbs up' if } X_{31} < 30 \text{ else 'thumbs down'}
\]
\[ x_{10} = -0.31 \cdot x_{00} + 0.99 \cdot x_{01} + (-0.63) \]
\[ x_{11} = -1.25 \cdot x_{00} + (-0.64) \cdot x_{01} + 1.88 \]
\[ x_{10} = 0 \text{ if } x_{10} < 0 \text{ else } x_{10} \]
\[ x_{11} = 0 \text{ if } x_{11} < 0 \text{ else } x_{11} \]
\[ x_{20} = 0.40 \cdot x_{10} + 1.21 \cdot x_{11} + 0.00 \]
\[ x_{21} = 0.64 \cdot x_{10} + 0.69 \cdot x_{11} + (-0.39) \]
\[ x_{20} = 0 \text{ if } x_{20} < 0 \text{ else } x_{20} \]
\[ x_{21} = 0 \text{ if } x_{21} < 0 \text{ else } x_{21} \]
\[ x_{30} = 0.26 \cdot x_{20} + 0.33 \cdot x_{21} + 0.45 \]
\[ X_{31} = 1.42 \cdot x_{20} + 0.40 \cdot x_{21} + (-0.45) \]
return ‘\(\text{👍}\)’ if \(X_{31} < 30\) else ‘\(\text{👎}\)’
\[x_{10} = -0.31 \times x_{00} + 0.99 \times x_{01} + (-0.63)\]
\[x_{11} = -1.25 \times x_{00} + (-0.64) \times x_{01} + 1.88\]
\[x_{10} = 0 \text{ if } x_{10} < 0 \text{ else } x_{10}\]
\[x_{11} = 0 \text{ if } x_{11} < 0 \text{ else } x_{11}\]
\[x_{20} = 0.40 \times x_{10} + 1.21 \times x_{11} + 0.00\]
\[x_{21} = 0.64 \times x_{10} + 0.69 \times x_{11} + (-0.39)\]
\[x_{20} = 0 \text{ if } x_{20} < 0 \text{ else } x_{20}\]
\[x_{21} = 0 \text{ if } x_{21} < 0 \text{ else } x_{21}\]
\[x_{30} = 0.26 \times x_{20} + 0.33 \times x_{21} + 0.45\]
\[X_{31} = 1.42 \times x_{20} + 0.40 \times x_{21} + (-0.45)\]
return ‘thumbs up’ if \(X_{31} < 30\) else ‘thumbs down’
\[
x_{10} = -0.31 \cdot x_{00} + 0.99 \cdot x_{01} + (-0.63)
\]
\[
x_{11} = -1.25 \cdot x_{00} + (-0.64) \cdot x_{01} + 1.88
\]
\[
x_{10} = 0 \text{ if } x_{10} < 0 \text{ else } x_{10}
\]
\[
x_{11} = 0 \text{ if } x_{11} < 0 \text{ else } x_{11}
\]
\[
x_{20} = 0.40 \cdot x_{10} + 1.21 \cdot x_{11} + 0.00
\]
\[
x_{21} = 0.64 \cdot x_{10} + 0.69 \cdot x_{11} + (-0.39)
\]
\[
x_{20} = 0 \text{ if } x_{20} < 0 \text{ else } x_{20}
\]
\[
x_{21} = 0 \text{ if } x_{21} < 0 \text{ else } x_{21}
\]
\[
x_{30} = 0.26 \cdot x_{20} + 0.33 \cdot x_{21} + 0.45
\]
\[
x_{31} = 1.42 \cdot x_{20} + 0.40 \cdot x_{21} + (-0.45)
\]
\[
x_{30} \geq x_{31} \quad x_{31} \geq x_{30}
\]
\[
\text{return } \text{ if } x_{31} < 30 \text{ else } '
\]
\[
L = 0.25
\]
\[
U = 2
\]
Nowadays, machine-learned software plays an increasingly important role in critical decision-making in our social, economic, and civic lives.
Formal Methods for Model Training
Robust Training
Minimizing the Worst-Case Loss for Each Input

Adversarial Training
Minimizing a Lower Bound on the Worst-Case Loss for Each Input
- generate adversarial inputs and use them as training data

Certified Training
Minimizing an Upper Bound on the Worst-Case Loss for Each Input
- use upper bound as regularizer to encourage robustness
Bibliography


Bibliography


