Scalable DNN verification using Constraint Solving

ThanhVu (Vu) Nguyen
DNN EVERYWHERE
DNN Problems

Amazon Rekognition FALSE MATCHES

28 current members of Congress
Nicolas Kayser-Bril
@nicolaskb

Black person with hand-held thermometer = firearm. Asian person with hand-held thermometer = electronic device.

Computer vision is so utterly broken it should probably be started over from scratch.
Robustness Properties

In a sentiment analysis task for medical records, with two misspelt words, a well-trained deep learning model is classified as a given label on DNNs are essentially caused by the inconsistency of the medical records – will lead to significant mis-classification on autonomous vehicles, by adding some natural transformations within a very small decision oracle

As shown in the second row of Fig. 4, a state-of-the-art DNN targeted local robustness means that a specific label can associate function.

Intuitively, local robustness states that all inputs in the region have the same class as input.

\[ \text{Definition 9} \]

Test Oracle of Local Robustness Property

**Definition 8**

Robustness requires that the decision of a DNN is invariant to small perturbations. The following definition is adapted
Safety Properties
DNN Verification

**Question**: Given a network $N$ and a property $p$, does $N$ have $p$?

- $p$ often has the form $P \Rightarrow Q$ (precondition $P$, postcondition $Q$)

**Answer**: Yes / No
DNN Verification

**Question:** Given a network $N$ and a property $p$, does $N$ have $p$?

- $p$ often has the form $P \Rightarrow Q$ (precondition $P$, postcondition $Q$)

**Answer:** Yes / No

![Diagram of a neural network]

- **Valid:** $x_1 \in [-1, 1] \land x_2 \in [-2, 2] \Rightarrow x_5 \leq 0$
- **Invalid:** $x_1 \in [-1, 1] \land x_2 \in [-2, 2] \Rightarrow x_5 < 0$
Abstraction

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval (ReluVal), zonotopes (ERAN), polytopes ($\alpha, \beta$-CROWN)

![Interval Diagram](image1)

![Zonotope Diagram](image2)

![Polytope Diagram](image3)
Abstraction

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval (ReluVal), zonotopes (ERAN), polytopes ($\alpha, \beta$-CROWN)

Scale well, but *loose precision* (producing spurious cex’s)
  - Newer work: iterative refine abstraction to filter spurious cex’s
Constraint Solving

Verification Query

Input Space | Neural Network | Output Space
---|---|---
\( P \) | | \( q \)

Veriﬁcation

\( \text{SAT (+ counter example)} \) | \( \text{UNSAT} \)

Transform DNN veriﬁcation into a constraint (satisfiability) problem

\[
\begin{align*}
P \quad \Rightarrow \quad \text{SAT} + \text{counter example} \quad \text{or} \quad \text{UNSAT}
\end{align*}
\]

MILP (Reluplex, Marabou)-based solvers

SMT solvers (Planet, DLV) or customized simplex

To prove

\[
\begin{align*}
\text{SAT}: & \quad \exists \vec{x} \in \mathbb{R}^n \quad \text{such that} \quad N_{\vec{x}}(\vec{N}) \quad \text{is satisfiable} \\
\text{UNSAT}: & \quad \forall \vec{x} \in \mathbb{R}^n \quad N_{\vec{x}}(\vec{N}) \quad \text{is unsatisﬁable}
\end{align*}
\]

HUGE
Transform DNN verification into a constraint (satisfiability) problem

- To prove $N \Rightarrow p$ (where $p$ is $P \Rightarrow Q$)
  - check if $\neg(N \Rightarrow (P \Rightarrow Q))$, i.e., $N \land P \land \neg Q$ is satisfiable
  - **UNSAT**: $p$ is a property of $N$
  - **SAT**: $p$ is not a property of $N$ (also give counterexample inputs satisfying $P$ but not $Q$)
Constraint Solving

Transform DNN verification into a constraint (satisfiability) problem

- To prove $N \Rightarrow p$ (where $p$ is $P \Rightarrow Q$)
  - check if $\neg(N \Rightarrow (P \Rightarrow Q))$, i.e., $N \land P \land \neg Q$ is satisfiable
  - **UNSAT**: $p$ is a property of $N$
  - **SAT**: $p$ is not a property of $N$ (also give counterexample inputs satisfying $P$ but not $Q$)

Solve the constraint(s)

- SMT solvers (Planet, DLV) or customized simplex
- MILP (Reluplex, Marabou)-based solvers

Scalability is a **HUGE** problem
Complexity and Scalability

Complexity: NP-Complete

- **Scalability** is the main problem
- State-of-the-art verification tools: networks with $138M$ of parameters, 160K inputs
- Real-world networks: $3.5B$ parameters, 1.2M of inputs
NeuralSAT: Our DNN Constraint Solver

Use NeuralSAT to prove $N \Rightarrow (P \Rightarrow Q)$

- Call NeuralSAT($N \land P \land \neg Q$)
- Return UNSAT or SAT (and counterexample)

Insight: combines conflict clause learning in SAT solving and abstraction for scalability
Example

To prove $f : x_1 \in [-1, 1] \land x_2 \in [-2, 2] \Rightarrow x_5 \leq 0$:

- NeuralSAT($\neg f$) =
  NeuralSAT($\neg (N \land x_1 \in [-1, 1] \land x_2 \in [-2, 2] \land x_5 > 0)$)

- NeuralSAT returns UNSAT, indicating $f$ is valid
Boolean Abstraction

- Create 2 boolean variables \( v_3 \) and \( v_4 \) to represent activation status of \( x_3, x_4 \)
  - \( v_3 = T \) means \( x_3 \) is active,
  - \( -x_1 - 0.5x_2 - 1 > 0 \)

- Form two clauses \( \{ v_3 \lor \overline{v_3} ; v_4 \lor \overline{v_4} \} \)

- Find boolean values for \( v_3, v_4 \) that satisfies the clauses and their implications

\[
x_1 \in [-1, 1], \; x_2 \in [-2, 2], \; x_5 > 0
\]
Iteration 1

- Use **abstraction** to approximate upperbound $x_5 \leq 0.55$ (from $x_1 \in [-1, 1], x_2 \in [-2, 2]$)
- **Deduce** $x_5 > 0$ *might be* feasible
- **Decide** $v_3 = F$ (randomly)
  - new constraint $-x_1 - 0.5x_2 - 1 < 0$

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
Iteration 2

- **Approximate** upperbound $x_5 \leq 0$ (due to additional constraint from $v_3 = F$)
- **Deduce** $x_5 > 0$ not feasible: CONFLICT
- **Analyze** conflict, **backtrack** and erase prev. decision $v_3 = F$
- **Learn** new clause $v_3$
  - $v_3$ will have to be $T$ in next iteration

$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$
**Iteration 3**

- **Decide** $v_3 = T$ (**BCP**, due to learned clause $v_3$)
  - new constraint $-x_1 - 0.5x_2 - 1 > 0$
- **Approximate** new upperbound for $x_5$ (using additional constraint from $v_3 = T$)
- **Deduce** $x_5 > 0$ might be feasible
- **Decide** $v_4 = T$ (randomly)

\[ x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0 \]
After several iterations

- **Learn** clauses \( \{ v_3, \overline{v_3} \lor v_4, \overline{v_3} \lor \overline{v_4} \} \)
- **Deduce** not possible to satisfy the clauses
- **Return** **UNSAT**

- Cannot find inputs satisfying \( x_1 \in [-1, 1], x_2 \in [-2, 2] \) that cause \( N \) to return \( x_5 > 0 \)
- Hence, \( x_5 \leq 0 \) holds (i.e., the original property is valid)

\( x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0 \)
NeuralSAT’s Prototype and Preliminary Results

- Written in Python
- Accept standard DNN formats and specs
- Use DPLL/CDCL algorithms for clause learning and conflict analysis
- Use the polytope abstraction (can be replace with any other abstractions)
ACAS XU Results
Much faster than the constraint solver Marabou

<table>
<thead>
<tr>
<th>Prop</th>
<th>NeuralSAT</th>
<th>Marabou</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_1$</td>
<td>1025.36</td>
<td>TO (3 hrs)</td>
</tr>
<tr>
<td>$\phi_2^*$</td>
<td>22.84</td>
<td>821.41</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>526.77</td>
<td>8309.09</td>
</tr>
<tr>
<td>$\phi_4$</td>
<td>330.83</td>
<td>133.97</td>
</tr>
<tr>
<td>$\phi_5$</td>
<td>83.51</td>
<td>1259.74</td>
</tr>
<tr>
<td>$\phi_6$</td>
<td>127.35</td>
<td>250.41</td>
</tr>
<tr>
<td>$\phi_7^*$</td>
<td>262.01</td>
<td>TO</td>
</tr>
<tr>
<td>$\phi_8^*$</td>
<td>0.15</td>
<td>TO</td>
</tr>
<tr>
<td>$\phi_9$</td>
<td>142.00</td>
<td>TO</td>
</tr>
<tr>
<td>$\phi_{10}$</td>
<td>191.99</td>
<td>3134.35</td>
</tr>
</tbody>
</table>

Promising because NeuralSAT is a prototype with no optimizations. Still much slower than the abstraction tool nnenum which applies a series of 7 optimizations.
ACAS XU Results
Much faster than the constraint solver Marabou

<table>
<thead>
<tr>
<th>Prop</th>
<th>NeuralSAT</th>
<th>Marabou</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_1$</td>
<td>1025.36</td>
<td>TO (3 hrs)</td>
</tr>
<tr>
<td>$\phi_2^*$</td>
<td>22.84</td>
<td>821.41</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>526.77</td>
<td>8309.09</td>
</tr>
<tr>
<td>$\phi_4$</td>
<td>330.83</td>
<td>133.97</td>
</tr>
<tr>
<td>$\phi_5$</td>
<td>83.51</td>
<td>1259.74</td>
</tr>
<tr>
<td>$\phi_6$</td>
<td>127.35</td>
<td>250.41</td>
</tr>
<tr>
<td>$\phi_7^*$</td>
<td>262.01</td>
<td>TO</td>
</tr>
<tr>
<td>$\phi_8^*$</td>
<td>0.15</td>
<td>TO</td>
</tr>
<tr>
<td>$\phi_9$</td>
<td>142.00</td>
<td>TO</td>
</tr>
<tr>
<td>$\phi_{10}$</td>
<td>191.99</td>
<td>3134.35</td>
</tr>
</tbody>
</table>

- Promising because NeuralSAT is a prototype with no optimizations

Still much slower than the abstraction tool nnenum

- nnenum applies a series of 7 optimizations
- comparable if nnenum runs using single thread
Current Work / Future Directions

Current optimizations for NeuralSAT

- Parallize algorithms (e.g., Branch and Bound)
- Develop more precise (but still fast) abstraction
- Different search heuristics for boolean decisions
Current Work / Future Directions

Current optimizations for NeuralSAT

- Parallize algorithms (e.g., Branch and Bound)
- Develop more precise (but still fast) abstraction
- Different search heuristics for boolean decisions

Future Directions

- Support richer specifications
- Mining specifications
- Apply formal reasoning (verification, specs. mining) to GNNs