Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks

September 9, 2022

目录

目录

Contribution

- This technique is based on the simplex method, extended to handle the non-convex Rectified Linear Unit (ReLU) activation function
- They evaluated this technique on a prototype deep neural network implementation of the next-generation airborne collision avoidance system for unmanned aircraft (ACAS Xu).

Background

SMT(Satisfiability Modulo Theories):

- A **Boolean satisfiability (SAT)** engine operates on a Boolean abstraction of the formula, performing Boolean propagation, case-splitting, and Boolean conflict resolution.
- A **theory solver** can determine the satisfiability of theory.

Background

Linear Real Arithmetic $\mathcal{T}_{\mathbb{R}}$:

- $\mathcal{T}_{\mathbb{R}}$ consists of the signature containing all rational number constants and the symbols $\{+, -, \cdot, \leq, \geq\}$,paired with the standard model of the real numbers
- **linear formulas**: scalar-multiplication

$$\sum_{x_i \in \mathcal{X}} c_i x_i \bowtie d \tag{1}$$

for $\bowtie \in \{=, \leq, \geq\}$, $d \in \mathbb{R}$

Simplex: a standard and highly efficient decision procedure for determining the T_R-satisfiability of conjunctions of linear atoms.

定义 (Simplex configuration)

Simplex configuration is either {SAT,UNSAT} or a five-tuples $\langle \mathcal{B}, T, l, u, \alpha \rangle$:

- $\mathcal{B} \subseteq \mathcal{X}$: basic-variables set
- T: tableau, contains for each $x_i \in \mathcal{B}$ an equation $x_i = \sum_{x_j \notin \mathcal{B}} c_j x_j$
- l, u: lower and upper for every variables
- α: assignment, maps each variable $x \in \mathcal{X}$ to a real value

Steps of simplex algorithm:

initial configuration:

- for each item $\sum_{x_i \in \mathcal{X}} c_i x_i \bowtie d$, a new basic variable b is introduced, the equation $b = \sum_{x_i \in \mathcal{X}} c_i x_i$ is added to the tableau
- then d is added as a bound for b
- initial assignment: 0

modify the assignments of variables: first pivot

- T: regarded as a matrix expressing each of the basic variables (variables in $\mathcal{B}$) as a linear combination of non-basic variables (variables in $\mathcal{X}/\mathcal{B}$)
- $pivot(T, i, j)$: $x_i = \sum_{x_k \notin \mathcal{B}} c_k x_k \rightarrow x_j = \frac{x_i}{c_j} - \sum_{x_k \notin \mathcal{B}, k \neq j} \frac{c_k}{c_j} x_k$
- pivot step is switching of a basic variable x_i (the leaving variable) with a non-basic variable x_j (the entering variable)

modify the assignments of variables: then update (α, x_j, δ)

- non-basic variables: $\alpha'(x_j) = \alpha(x_j) + \delta$,else unchanged
- basic variables: $\alpha'(x_i) = \alpha(x_i) + \delta \cdot T_{i,j}$

$$\text{Pivot}_1 \quad \frac{x_i \in \mathcal{B}, \quad \alpha(x_i) < l(x_i), \quad x_j \in \text{slack}^+(x_i)}{T := pivot(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\text{Pivot}_2 \quad \frac{x_i \in \mathcal{B}, \quad \alpha(x_i) > u(x_i), \quad x_j \in \text{slack}^-(x_i)}{T := pivot(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

$$\text{Update} \quad \frac{x_j \notin \mathcal{B}, \quad \alpha(x_j) < l(x_j) \vee \alpha(x_j) > u(x_j), \quad l(x_j) \leq \alpha(x_j) + \delta \leq u(x_j)}{\alpha := update(\alpha, x_j, \delta)}$$

$$\text{slack}^+(x_i) = \{x_j \notin \mathcal{B} \mid (T_{i,j} > 0 \wedge \alpha(x_j) < u(x_j)) \vee (T_{i,j} < 0 \wedge \alpha(x_j) > l(x_j))$$
$$\text{slack}^-(x_i) = \{x_j \notin \mathcal{B} \mid (T_{i,j} < 0 \wedge \alpha(x_j) < u(x_j)) \vee (T_{i,j} > 0 \wedge \alpha(x_j) > l(x_j))$$

- slack: x_j affords slack $\rightarrow$ $x_i$closer to bound
- $+, -$:assignment increse or decrease

$$\text{Failure}\quad \frac{x_i \in \mathcal{B},\quad (\alpha(x_i) < l(x_i) \ \wedge\ \text{slack}^+(x_i) = \emptyset) \vee (\alpha(x_i) > u(x_i) \ \wedge\ \text{slack}^-(x_i) = \emptyset)}{\texttt{UNSAT}}$$

$$\text{Success}\quad \frac{\forall x_i \in \mathcal{X}.\ l(x_i) \leq \alpha(x_i) \leq u(x_i)}{\texttt{SAT}}$$

Simplex allows variables to temporarily violate their bounds as it iteratively looks for a feasible variable assignment

目录

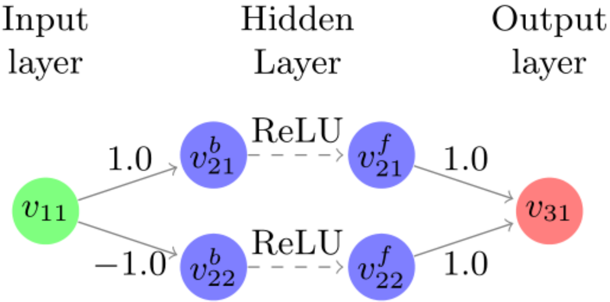
Relu + Linear Real Arithmetic

How to adapt simplex to DNN

- n nodes $\rightarrow 2^n$ conditions
- this theoretical worst-case behavior is also seen in practice

$\mathcal{T}_{\mathbb{R}} \to \mathcal{T}_{\mathbb{R}R}$:

- Signature: additionally includes the binary predicate **ReLU** with the interpretation: $ReLU(x, y) \iff y = \max(0, x)$
- Formulas: either linear inequalities or applications of the ReLU predicate to linear terms



Modify Relu nodes:

- Relu node $v \rightarrow v_b, v_f$
- assert $Relu(v_b, v_f)$

Reluplex

- Reluplex also allows variables that are members of ReLU pairs to temporarily violate the ReLU semantics.
- and corrects them using **Pivot** and **Update** operations.

定义 (Reluplex configuration)

Reluplex configuration is either {SAT,UNSAT} or a six-tuples $\langle \mathcal{B}, T, l, u, \alpha, R \rangle$:

- $\mathcal{B}, T, l, u, \alpha$ are as Simplex configuration
- $R \subset \mathcal{X} \times \mathcal{X}$ is the set of ReLU connections.

The **initial configuration** for a conjunction of atoms is also obtained as before except that $\langle x, y \rangle \in R$ iff $ReLU(x, y)$ is an atom.

Reluplex

- Reluplex includes simplex transition rules **Pivot1**, **Pivot2** and **Update**, which are designed to handle out-of-bounds violations
- **Success** relu → **ReluSuccess** relu

$$\textsf{ReluSuccess}\quad \frac{\forall x \in \mathcal{X}.\ l(x) \leq \alpha(x) \leq u(x), \quad \forall \langle x^b, x^f \rangle \in R.\ \alpha(x^f) = \max(0, \alpha(x^b))}{\texttt{SAT}}$$

Handling ReLU violations

$$\text{Update}_b \quad \frac{x_i \notin \mathcal{B}, \quad \langle x_i, x_j \rangle \in R, \quad \alpha(x_j) \neq \max(0, \alpha(x_i)), \quad \alpha(x_j) \geq 0}{\alpha := update(\alpha, x_i, \alpha(x_j) - \alpha(x_i))}$$

$$\text{Update}_f \quad \frac{x_j \notin \mathcal{B}, \quad \langle x_i, x_j \rangle \in R, \quad \alpha(x_j) \neq \max(0, \alpha(x_i))}{\alpha := update(\alpha, x_j, \max(0, \alpha(x_i)) - \alpha(x_j))}$$

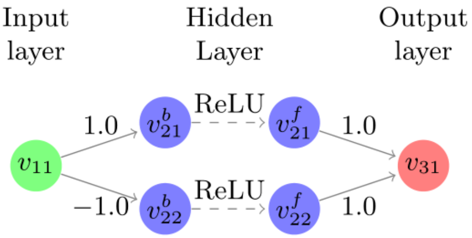
$$\text{PivotForRelu} \quad \frac{x_i \in \mathcal{B}, \quad \exists x_l.\ \langle x_i, x_l \rangle \in R \vee \langle x_l, x_i \rangle \in R, \quad x_j \notin \mathcal{B}, \quad T_{i,j} \neq 0}{T := pivot(T, i, j), \quad \mathcal{B} := \mathcal{B} \cup \{x_j\} \setminus \{x_i\}}$$

- The $Update_b$ and $Update_f$ rules allow a broken ReLU connection to be corrected by updating the backward or forward-facing variables respectively
- The **PivotForRelu** rule allows a basic variable appearing in a ReLU to be pivoted so that either $Update_b$ or $Update_f$ can be applied

ReluSplit

- Split relu: **ReluSplit** $\dfrac{\langle x_i, x_j\rangle \in R, \quad l(x_i) < 0, \quad u(x_i) > 0}{u(x_i) := 0 \ \vee\ l(x_i) := 0}$
- In practice, splitting can be managed by a SAT engine with **splitting-on-demand**
- Naive approach: applying the ReluSplit rule eagerly until it no longer applies and then solving each derived sub-problem separately
- Efficient approach: keep updating until the number of updates to a specific ReLU pair exceeds some threshold

Example



- Verify: $v_{11} \in [0,1]$ and $v_{31} \in [0.5,1]$
- initial Relu configuration:

$$\begin{cases} a_1 = -v_{11} + v_{21}^b \\ a_2 = v_{11} + v_{22}^b \\ a_3 = -v_{21}^f - v_{22}^f + v_{31} \end{cases} \tag{2}$$

- $\mathcal{B} = \{a_1, a_2, a_3\}$, $R = \left\{\left\langle v_{21}^b, v_{21}^f \right\rangle, \left\langle v_{22}^b, v_{22}^f \right\rangle\right\}$
- initial assignment:0; l and u of $\mathcal{B}$: 0
- hidden variables:unbound,except v_f are non negative

Example

variable	v_{11}	v_{21}^b	v_{21}^f	v_{22}^b	v_{22}^f	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0	0	0	0	0	0	0	0	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

variable	v_{11}	v_{21}^b	v_{21}^f	v_{22}^b	v_{22}^f	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0	0	0.5	0	0	0.5	0	0	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

First: fix any out-of-bounds variables.

- Update: $\alpha(v_{31}) : 0 \to 0.5, \alpha(a_3) : 0 \to 0.5$
- pivot: $a_3 = -v_{21}^f - v_{22}^f + v_{31} \to v_{21}^f = -v_{22}^f + v_{31} - a_3$
- then update: $\alpha(a_3) : 0.5 \to 0, \alpha(v_{21}^f) : 0 \to 0.5$
- now tableau:

$$\begin{cases} a_1 = -v_{11} + v_{21}^b \\ a_2 = v_{11} + v_{22}^b \\ v_{21}^f = -v_{22}^f + v_{31} - a_3 \end{cases} \tag{3}$$

Example

variable	v_{11}	v^b_{21}	v^f_{21}	v^b_{22}	v^f_{22}	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0	0.5	0.5	0	0	0.5	0.5	0	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

variable	v_{11}	v^b_{21}	v^f_{21}	v^b_{22}	v^f_{22}	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0.5	0.5	0.5	0	0	0.5	0	0.5	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

Then: fix the broken ReLU pair.

- $\alpha\left(v^f_{21}\right) = 0.5 \neq 0 = \max\left(0, \alpha\left(v^b_{21}\right)\right)$
- $Update_b$: $\alpha(v^b_{21}) : 0 \to 0.5, \alpha(a_1) : 0 \to 0.5$
- $\alpha(a_1)$ is out of bound.
- pivot: $a_1 = -v_{11} + v^b_{21} \to v_{11} = v^b_{21} - a_1, a_2 = v_{11} + v^b_{22} \to a_2 = v^b_{21} + v^b_{22} - a_1$
- update: $\alpha(a_1) : 0.5 \to 0, \alpha(v_{11}) : 0 \to 0.5, \alpha(a_2) : 0 \to 0.5$

$$\begin{cases} v_{11} = v^b_{21} - a_1 \\ a_2 = v^b_{21} + v^b_{22} - a_1 \\ v^f_{21} = -v^f_{22} + v_{31} - a_3 \end{cases} \tag{4}$$

Example

variable	v_{11}	v_{21}^b	v_{21}^f	v_{22}^b	v_{22}^f	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0.5	0.5	0.5	0	0	0.5	0	0.5	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

variable	v_{11}	v_{21}^b	v_{21}^f	v_{22}^b	v_{22}^f	v_{31}	a_1	a_2	a_3
lower bound	0	$-\infty$	0	$-\infty$	0	0.5	0	0	0
assignment	0.5	0.5	0.5	-0.5	0	0.5	0	0	0
upper bound	1	∞	∞	∞	∞	1	0	0	0

Iteratively fix out-of-bounds variables.

- $\alpha(a_2)$ is out of bound.
- pivot:
  $a_2 = v_{21}^b + v_{22}^b - a_1 \rightarrow v_{22}^b = -v_{21}^b + a_1 + a_2$
- update:
  $\alpha(a_2) : 0.5 \rightarrow 0, \alpha(v_{22}^b) : 0 \rightarrow -0.5$

$$\begin{cases} v_{11} = v_{21}^b - a_1 \\ v_{22}^b = -v_{21}^b + a_1 + a_2 \\ v_{21}^f = -v_{22}^f + v_{31} - a_3 \end{cases} \tag{5}$$

目录

Tighter Bound Derivation

- $\text{pos}(x_i) = \{x_j \notin \mathcal{B} \mid T_{i,j} > 0\}, \text{neg}(x_i) = \{x_j \notin \mathcal{B} \mid T_{i,j} < 0\}$
- Throughout the execution, the following rules can be used to derive tighter bounds for x_i, regardless of the current assignment:

$$\text{deriveLowerBound}\ \frac{x_i \in \mathcal{B}, \quad l(x_i) < \sum_{x_j \in \text{pos}(x_i)} T_{i,j} \cdot l(x_j) + \sum_{x_j \in \text{neg}(x_i)} T_{i,j} \cdot u(x_j)}{l(x_i) := \sum_{x_j \in \text{pos}(x_i)} T_{i,j} \cdot l(x_j) + \sum_{x_j \in \text{neg}(x_i)} T_{i,j} \cdot u(x_j)}$$

$$\text{deriveUpperBound}\ \frac{x_i \in \mathcal{B}, \quad u(x_i) > \sum_{x_j \in \text{pos}(x_i)} T_{i,j} \cdot u(x_j) + \sum_{x_j \in \text{neg}(x_i)} T_{i,j} \cdot l(x_j)}{u(x_i) := \sum_{x_j \in \text{pos}(x_i)} T_{i,j} \cdot u(x_j) + \sum_{x_j \in \text{neg}(x_i)} T_{i,j} \cdot l(x_j)}$$

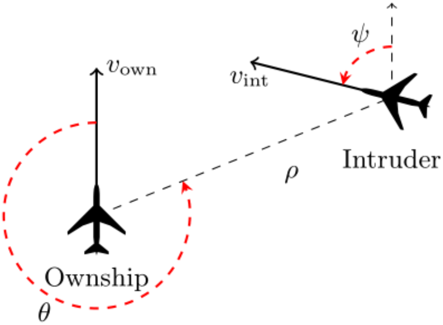
- contribute to fix the active or inactive state, without splitting

Derived Bounds and Conflict Analysis

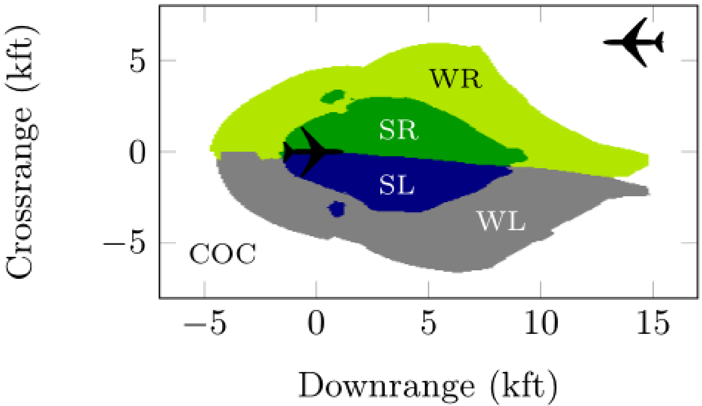
- Bound derivation can lead to situations: $l(x) > u(x)$
- Such contradictions allow Reluplex to immediately undo a previous split (or answer **UNSAT** if no previous splits exist).
- Use *Conflict Analysis*: in many cases more than just the previous split can be undone

目录

ACAS Xu(Airborne Collision Avoidance System X unmanned)



Prove Network Properties



Evaluation

Search strategy:

- First repeatedly fix any out-of-bounds violations
- Then correct any violated ReLU constraints(possibly introducing new out-of-bounds violations)
- Perform bound tightening on the entering variable after every pivot operation,and perform a more thorough bound tightening on all the equations in the tableau once every few thousand pivot steps.

Results

	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	φ_8
CVC4	-	-	-	-	-	-	-	-
Z3	-	-	-	-	-	-	-	-
Yices	1	37	-	-	-	-	-	-
MathSat	2040	9780	-	-	-	-	-	-
Gurobi	1	1	1	-	-	-	-	-
Reluplex	8	2	7	7	93	4	7	9

Results

	Networks	Result	Time	Stack	Splits
ϕ_1	41	UNSAT	394517	47	1522384
	4	TIMEOUT			
ϕ_2	1	UNSAT	463	55	88388
	35	SAT	82419	44	284515
ϕ_3	42	UNSAT	28156	22	52080
ϕ_4	42	UNSAT	12475	21	23940
ϕ_5	1	UNSAT	19355	46	58914
ϕ_6	1	UNSAT	180288	50	548496
ϕ_7	1	TIMEOUT			
ϕ_8	1	SAT	40102	69	116697
ϕ_9	1	UNSAT	99634	48	227002
ϕ_{10}	1	UNSAT	19944	49	88520

- Stack:maximal depth of nested case-splits reached
- Splits:total number of case-splits performed

Robustness

Table 3: Local adversarial robustness tests. All times are in seconds.

	$\delta = 0.1$		$\delta = 0.075$		$\delta = 0.05$		$\delta = 0.025$		$\delta = 0.01$		Total
	Result	Time	Result	Time	Result	Time	Result	Time	Result	Time	Time
Point 1	SAT	135	SAT	239	SAT	24	UNSAT	609	UNSAT	57	1064
Point 2	UNSAT	5880	UNSAT	1167	UNSAT	285	UNSAT	57	UNSAT	5	7394
Point 3	UNSAT	863	UNSAT	436	UNSAT	99	UNSAT	53	UNSAT	1	1452
Point 4	SAT	2	SAT	977	SAT	1168	UNSAT	656	UNSAT	7	2810
Point 5	UNSAT	14560	UNSAT	4344	UNSAT	1331	UNSAT	221	UNSAT	6	20462

- δ-*locally-robust* at input point x
- ϵ-*globally-robust*:only on small networks;improving the scalability of this technique is left for future work.

Thank you