Improving the Reliability and Safety of Systems

Toward Scalable Deep Neural Network Verification

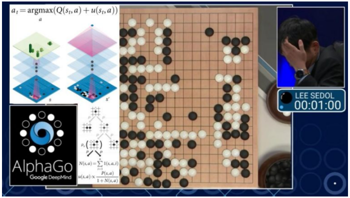
ThanhVu (Vu) Nguyen

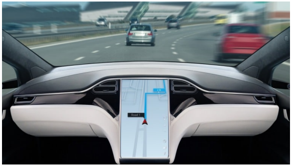
GEORGE MASON UNIVERSITY

SWE 619, Mar 20, 2024

Outline

AI Safety Verification





DNN EVERYWHERE

DNN Problems

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.

Screenshot from 2020-03-31 11-23-45.png

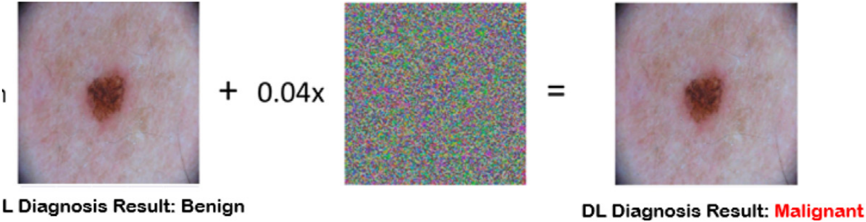
Gun	88%
Photography	68%
Firearm	65%
Plant	59%

Screenshot from 2020-03-31 11-27-22.png

Technology	68%
Electronic Device	66%
Photography	62%
Mobile Phone	54%

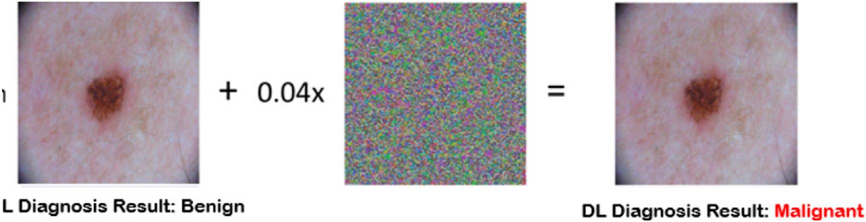

Happening NOW
GOOGLE SELF-DRIVING CAR GETS INTO AN ACCIDENT INVOLVING INJURIES
#NEWS
GOOGLE SELF DRIVING CAR
CRASHES INTO A BUS
EXCLUSIVE
TAKING ACTION FOR YOU
INVESTIGATION FOCUSED ON TESLA AUTOPILOT
abc ACTION NEWS
Mark Beach
TEMPE
DEADLY CRASH WITH SELF-DRIVING UBER
abc 15 ARIZONA
11:01
NEW VIDEO
TAKING ACTION
DRIVERLESS UBER CAR INVOLVED IN CRASH IN TEMPE
POLICE SAY OTHER DRIVER FAILED TO YIELD
abc 15 ARIZONA
6:06 81°

Robustness Properties



$$\forall i \in \{0 \ldots |X| - 1\}.\ X_i - Y_i \leq 0.1 \ \Rightarrow \ \mathit{class}(X) \equiv \mathit{class}(Y) \tag{1}$$

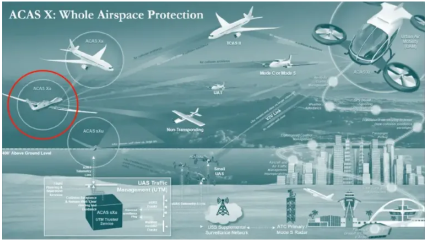
Robustness Properties



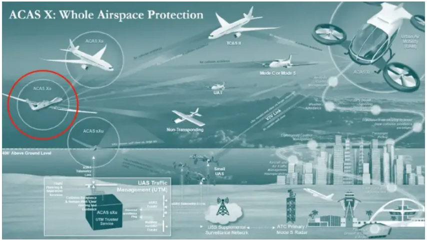
$$\forall i \in \{0 \ldots |X| - 1\}.\ X_i - Y_i \leq 0.1 \;\Rightarrow\; \mathit{class}(X) \equiv \mathit{class}(Y) \qquad (1)$$

if corresponding pixels of two images X and Y are not different by more than 0.1, then X and Y should have the same classification

Safety Properties



Safety Properties



ACAS: air traffic collision system, detects intruder and decides action.

$$d_{intru} \geq 55947 \wedge v_{own} \geq 1145 \wedge v_{intru} \leq 60 \;\Rightarrow\; r_{nothing} \leq \tau$$

if intruder is distant and significantly slower than us, then we do nothing (i.e., below a certain threshold)

DL Classification: Green Light

DL Classification: Red Light

- Well-trained, e.g., 97% accuracy, DNNs are fine for most tasks
  - But not enough for mission-critical tasks, e.g., self-driving cars, air traffic collision control
- Testing can find counterexamples (e.g., adversarial attacks)
  - Testing shows the existence of errors, not its absence (*Dijkstra*)

DL Classification: Green Light

DL Classification: Red Light

- Well-trained, e.g., 97% accuracy, DNNs are fine for most tasks
  - But not enough for mission-critical tasks, e.g., self-driving cars, air traffic collision control
- Testing can find counterexamples (e.g., adversarial attacks)
  - Testing shows the existence of errors, not its absence (*Dijkstra*)

Formal Verification Can Help!

Software Verification

- Provide formal guarantee that a system really has no specific type of errors
- Mature field in CS/Logics with lots of powerful techniques and tools
  - Automated Theorem Proving
  - Constraint Solving (e.g., SAT/SMT solving)
  - Model Checking
  - Abstract Interpretation, ...
- Employed in mission-critical systems, e.g., avionics, medical devices, Windows, Clouds system (AWS)

The problem of Deep Neural Network 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

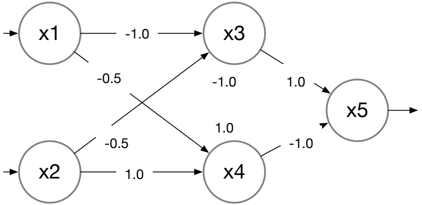
The problem of Deep Neural Network verification

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

Simple DNN with **ReLU**



- E.g., $x_3 = \max(-1x_1 + -0.5x_2, 0)$

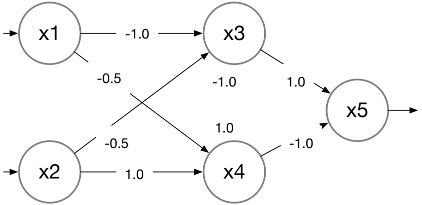
The problem of Deep Neural Network verification

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)

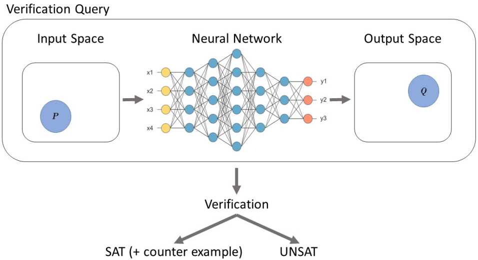
Answer: Yes / No

Simple DNN with **ReLU**

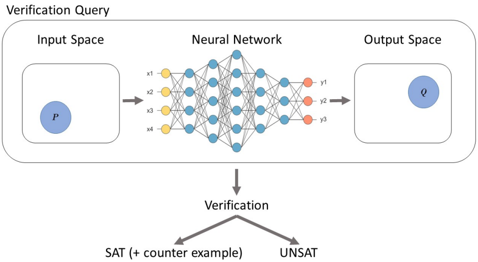


- E.g., $x_3 = \max(-1x_1 + -0.5x_2, 0)$
- Valid: $x_1 \in [-1, 1] \wedge x_2 \in [-2, 2] \Rightarrow x_5 \leq 0$
- Invalid: $x_1 \in [-1, 1] \wedge x_2 \in [-2, 2] \Rightarrow x_5 > 0$

Constraint Solving Techniques

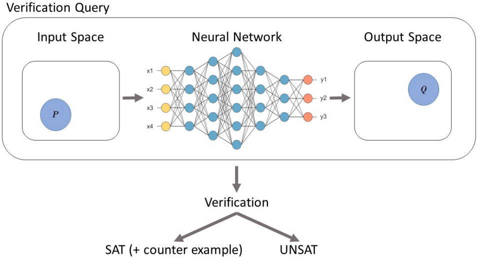


Constraint Solving Techniques



- Transform DNN verification into a constraint (satisfiability) problem
  - UNSAT: p is a property of N
  - SAT: p is not a property of N (also provide counterexamples)
  - TIMEOUT

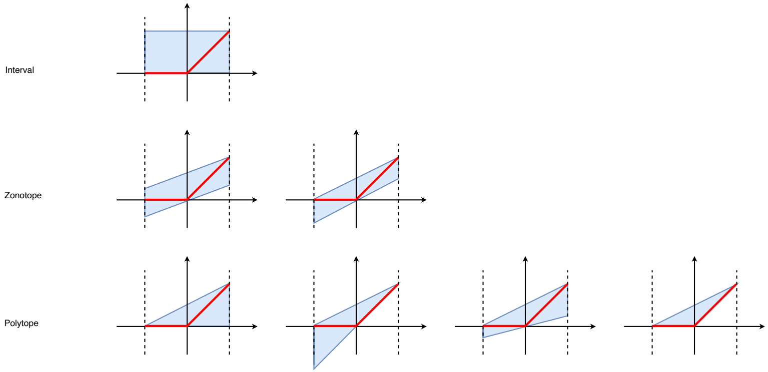
Constraint Solving Techniques



- Transform DNN verification into a constraint (satisfiability) problem
  - UNSAT: p is a property of N
  - SAT: p is not a property of N (also provide counterexamples)
  - TIMEOUT
- Solve the constraint, e.g., using MILP solvers
- Scalability is a Huge problem (many TIMEOUTs)
  - Complexity $O(2^N)$, where N is the number of neurons

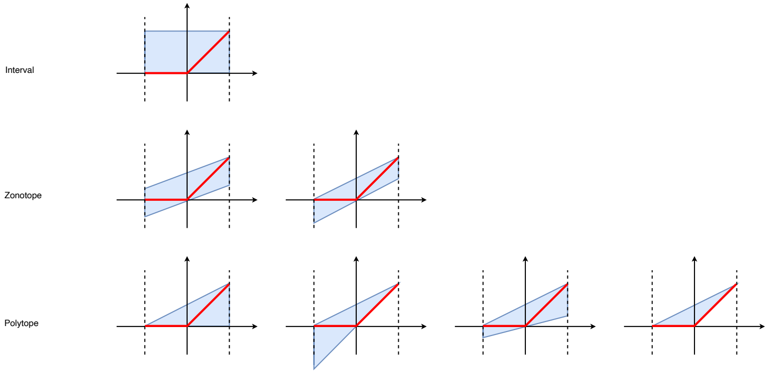
Abstraction Techniques

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval, zonotopes, polytopes



Abstraction Techniques

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval, zonotopes, polytopes

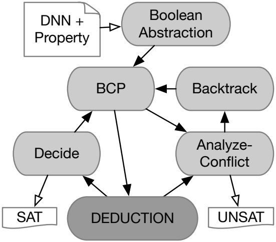


- Scale well, but loose precision (producing spurious cex's)
  - Claiming a property is violated when it is not

NeuralSAT: Our DNN Constraint Solver

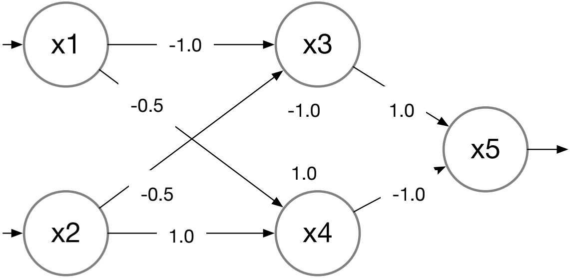
To prove $N \Rightarrow (P \Rightarrow Q)$

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



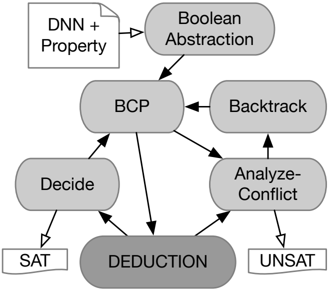
1. Abstract as a boolean satisfiability problem
2. Iteratively search for satisfying assignment
   - Use heuristics to make decision
   - Use propagation to communicate learn information
   - Analyze conflicts, learn conflict information, and backtrack
   - Use a theory solver to quickly deduce unsatisfiability (UNSAT)

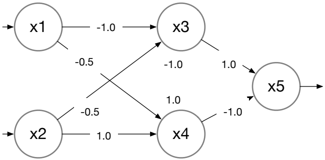
Example: Simple DNN with ReLU activation



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

- Use NeuralSAT to check if $\neg f$ is satisfiable
- NeuralSAT$(N \wedge x_1 \in [-1, 1] \wedge x_2 \in [-2, 2] \wedge x_5 > 0)$
- NeuralSAT returns UNSAT, indicating f is valid

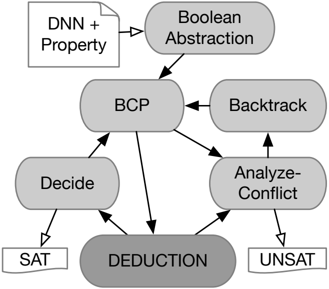


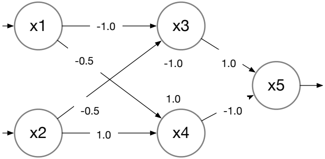


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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, $-x_1 - 0.5x_2 - 1 > 0$

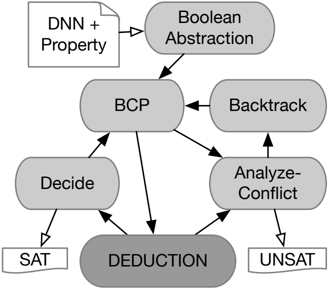


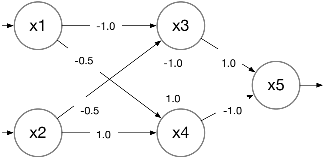


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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 $\{v_3 \vee \overline{v_3}\ ;\ v_4 \vee \overline{v_4}\}$
- Find boolean values for v_3, v_4 that satisfies the clauses and their implications

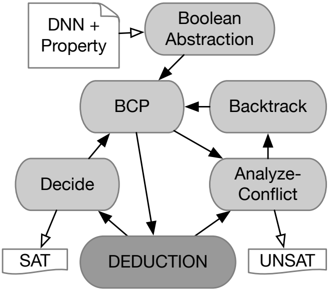


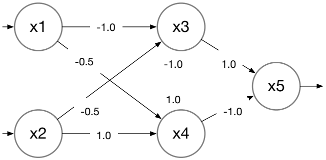


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

Iteration 1

- Use **abstraction** to approximate upperbound $x_5 \leq 0.55$ (from $x_1 \in [-1, 1], x_2 \in [-2, 2]$)

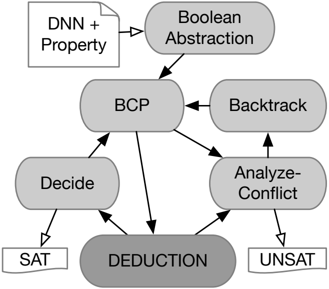


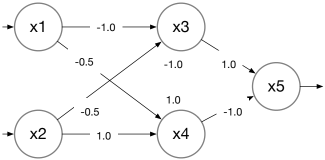


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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

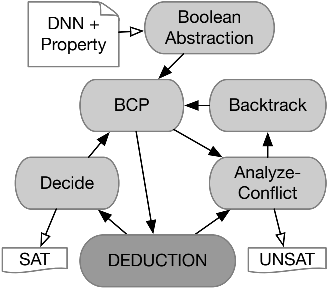


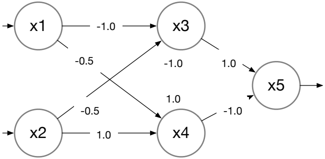


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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 $-x_1 - 0.5x_2 - 1 < 0$

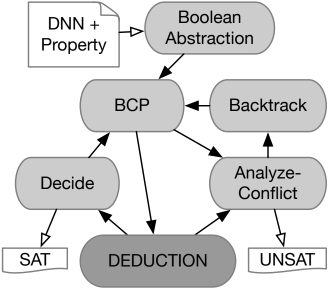


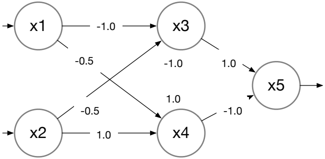


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

Iteration 2

- **Approximate** upperbound $x_5 \leq 0$ (due to additional constraint from $v_3 = F$)
- **Deduce** $x_5 > 0$ infeasible: CONFLICT

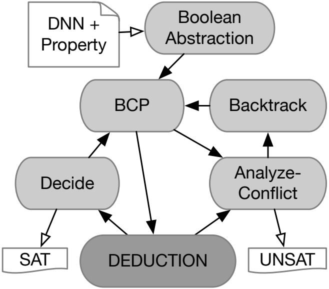


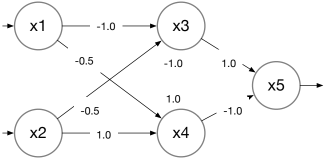


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

Iteration 2

- **Approximate** upperbound $x_5 \leq 0$ (due to additional constraint from $v_3 = F$)
- **Deduce** $x_5 > 0$ infeasible: 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

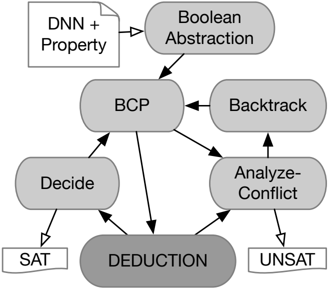


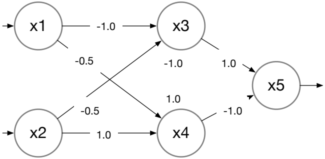


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

Iteration 3

- **Decide** $v_3 = T$ (**BCP**, due to learned clause v_3)
  - new constraint $-x_1 - 0.5x_2 - 1 > 0$

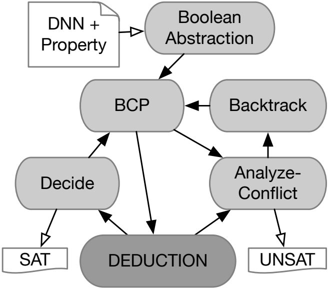


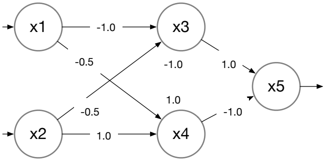


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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)
- $\vdots$

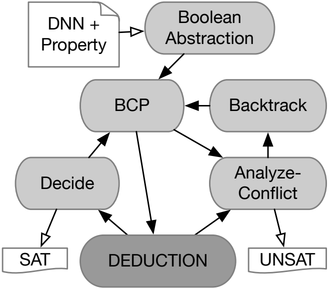


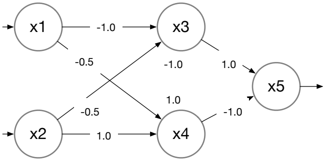


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

After several iterations

- **Learn** clauses $\{v_3, \overline{v_3} \vee v_4, \overline{v_3} \vee \overline{v_4}\}$
- **Deduce** not possible to satisfy the clauses





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

After several iterations

- **Learn** clauses $\{v_3, \overline{v_3} \vee v_4, \overline{v_3} \vee \overline{v_4}\}$
- **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)

Benchmark	Rank	Verifier	Score	Percent	Verify	Falsify
ACAS Xu (13K)	1	NeuralSAT	1437	100.0%	**139**	**47**
	1	nnenum	1437	100.0%	**139**	**47**
	3	$\alpha\beta$-CROWN	1436	99.9%	**139**	46
	4	Marabou	1426	99.2%	138	46
	5	MN-BaB	1097	76.3%	105	**47**
MNISTFC (532K)	1	$\alpha\beta$-CROWN	582	100.0%	**56**	22
	2	NeuralSAT	573	98.5%	55	**23**
	3	nnenum	403	69.2%	39	13
	4	MN-BaB	370	63.6%	36	10
	4	Marabou	370	63.6%	35	20
CIFAR2020 (2.5M)	1	NeuralSAT	1533	100.0%	**149**	**43**
	2	$\alpha\beta$-CROWN	1522	99.3%	148	42
	3	MN-BaB	1486	96.9%	145	36
	5	nnenum	518	33.8%	50	18
RESNET_AB (354K)	1	NeuralSAT	513	100.0%	**23**	**23**
	1	$\alpha\beta$-CROWN	513	100.0%	**49**	**23**
	3	MN-BaB	363	70.8%	34	**23**
MNIST_GDVB (3M)	1	NeuralSAT	480	100.0%	**48**	0
	2	$\alpha\beta$-CROWN	400	83.3%	40	0
	3	MN-BaB	200	41.7%	20	0
Overall	1	NeuralSAT	4536	100.0%	**440**	**136**
	2	$\alpha\beta$-CROWN	4453	98.2%	432	133
	3	MN-BaB	3516	77.5%	340	116
	4	nnenum	2358	52.0%	228	78
	5	Marabou	1796	39.6%	173	66

Key Ideas

- Formalization of DNN verification
- Analyze, learn, and propagate information (significantly reduce search space)
- Dedicated DNN-specific theory solver (enable fast proving)
- *New approach; open doors to new research on heuristics, optimizations specific to DNNs*

Key Ideas

- Formalization of DNN verification
- Analyze, learn, and propagate information (significantly reduce search space)
- Dedicated DNN-specific theory solver (enable fast proving)
- *New approach; open doors to new research on heuristics, optimizations specific to DNNs*

Usability Features

- Standard: inputs (ONNX) and outputs (SAT/UNSAT/TIMEOUT)
- Versatile
  - Support Feedforward, Convolutional, Residual Networks
  - Support ReLU, Sigmoid, Tanh, Power, etc
- Scale well to large networks with millions of neurons
- Active development & frequent Updates
- Fully automatic (require little configurations from users)