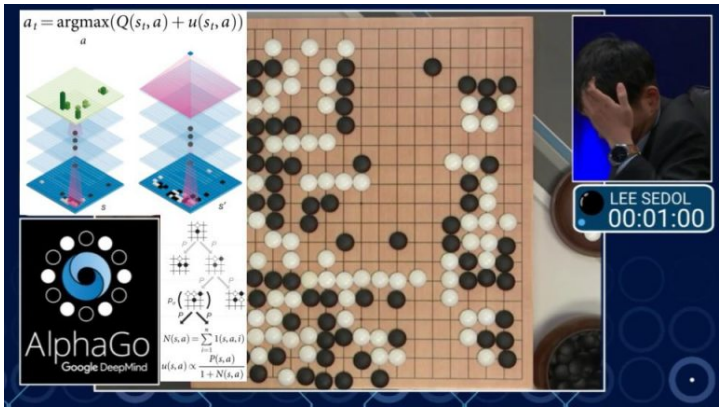
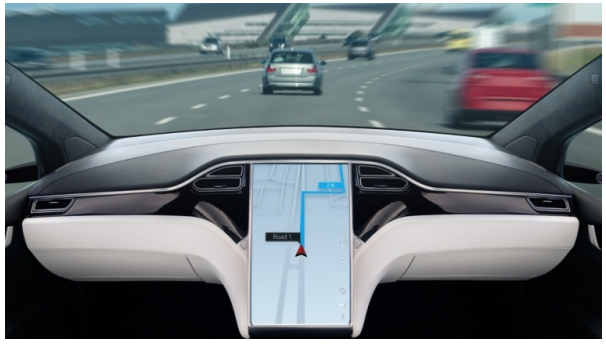
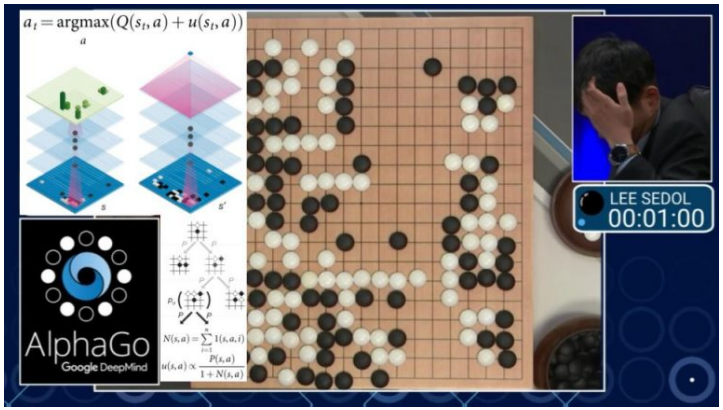
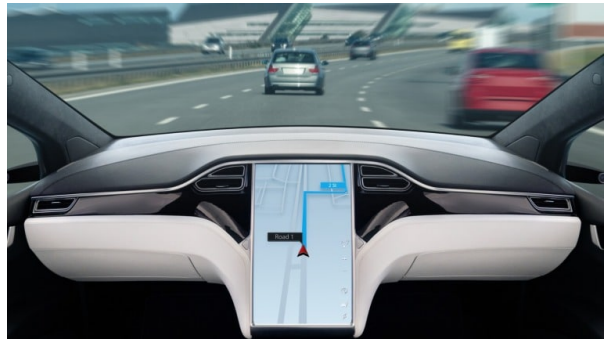
Scalable DNN verification using Constraint Solving

ThanhVu (Vu) Nguyen


$a_t = \underset{a}{\text{argmax}}(Q(s_t, a) + u(s_t, a))$
LEE SEDOL
00:01:00
AlphaGo
Google DeepMind
$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$
$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$







DNN EVERYWHERE

DNN Problems

Nicolas Kayser-Bril
@nicolaskb

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

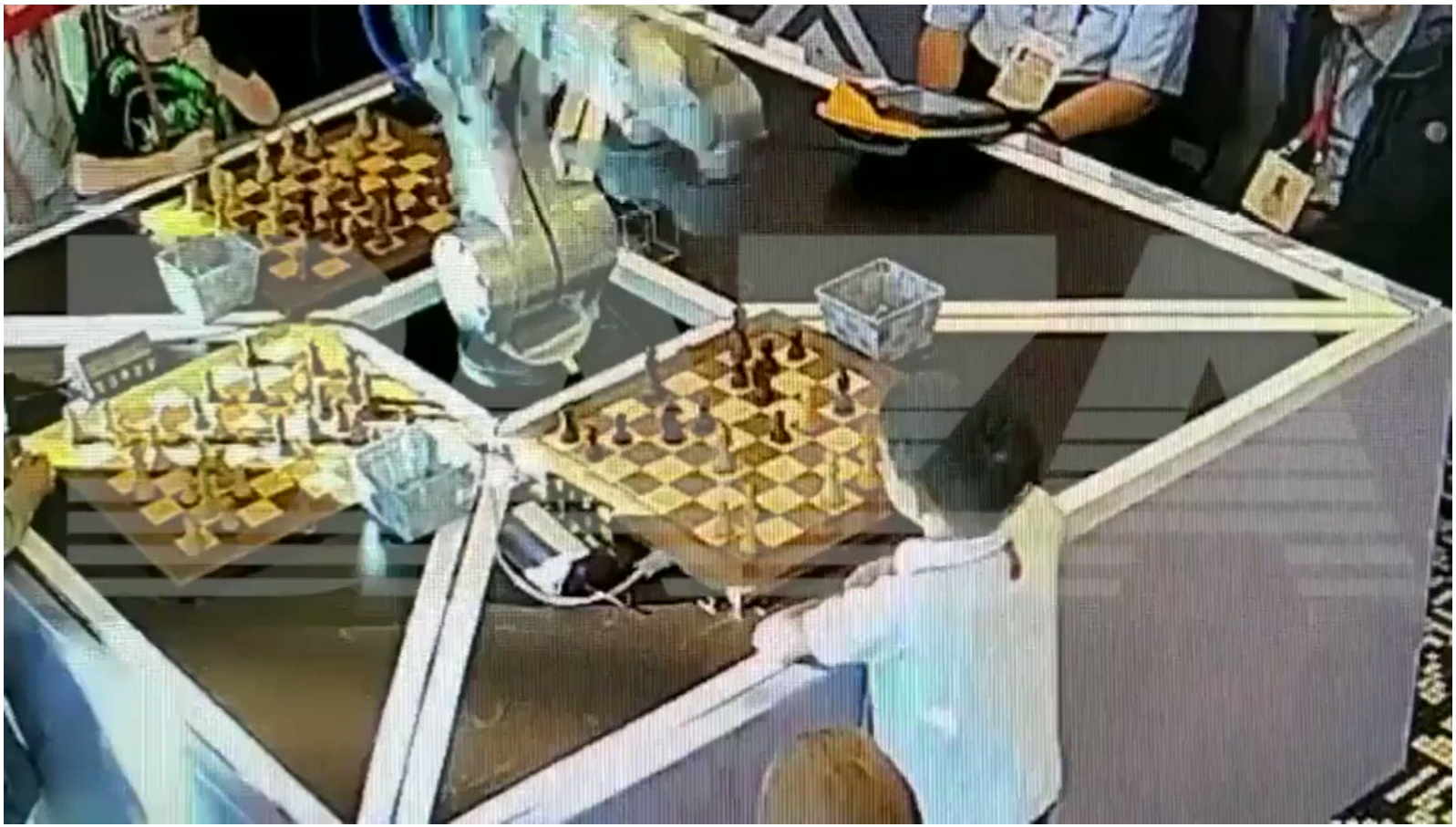
Computer vision is so utterly broken it should probably be started over from scratch.


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%




GOOGLE SELF-DRIVING CAR GETS INTO
AN ACCIDENT INVOLVING INJURIES
Happening NOW
EXCLUSIVE
TAKING ACTION FOR YOU
INVESTIGATION FOCUSED ON TESLA AUTOPILOT
ACTION NEWS
TEMPE
DEADLY CRASH WITH SELF-DRIVING UBER
ARIZONA
11:01


#NEWS
GOOGLE SELF DRIVING CAR
CRASHES INTO A BUS
Mark Beach
NEW VIDEO
TAKING ACTION
DRIVERLESS UBER CAR INVOLVED IN CRASH IN TEMPE
POLICE SAY OTHER DRIVER FAILED TO YIELD
ARIZONA
6:06

Robustness Properties

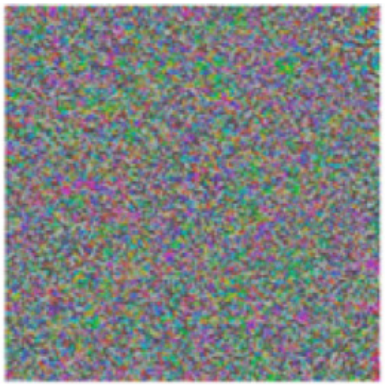
DL Classification: Green Light

Changing one pixel here

Text

DL Classification: Red Light

+ 0.04x



=

L Diagnosis Result: Benign

DL Diagnosis Result: Malignant

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)

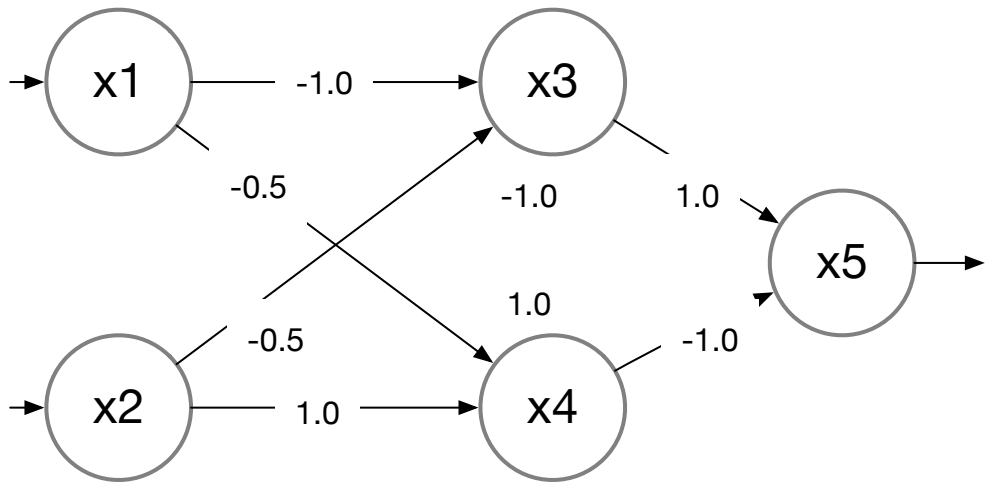
Answer: Yes / No

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)

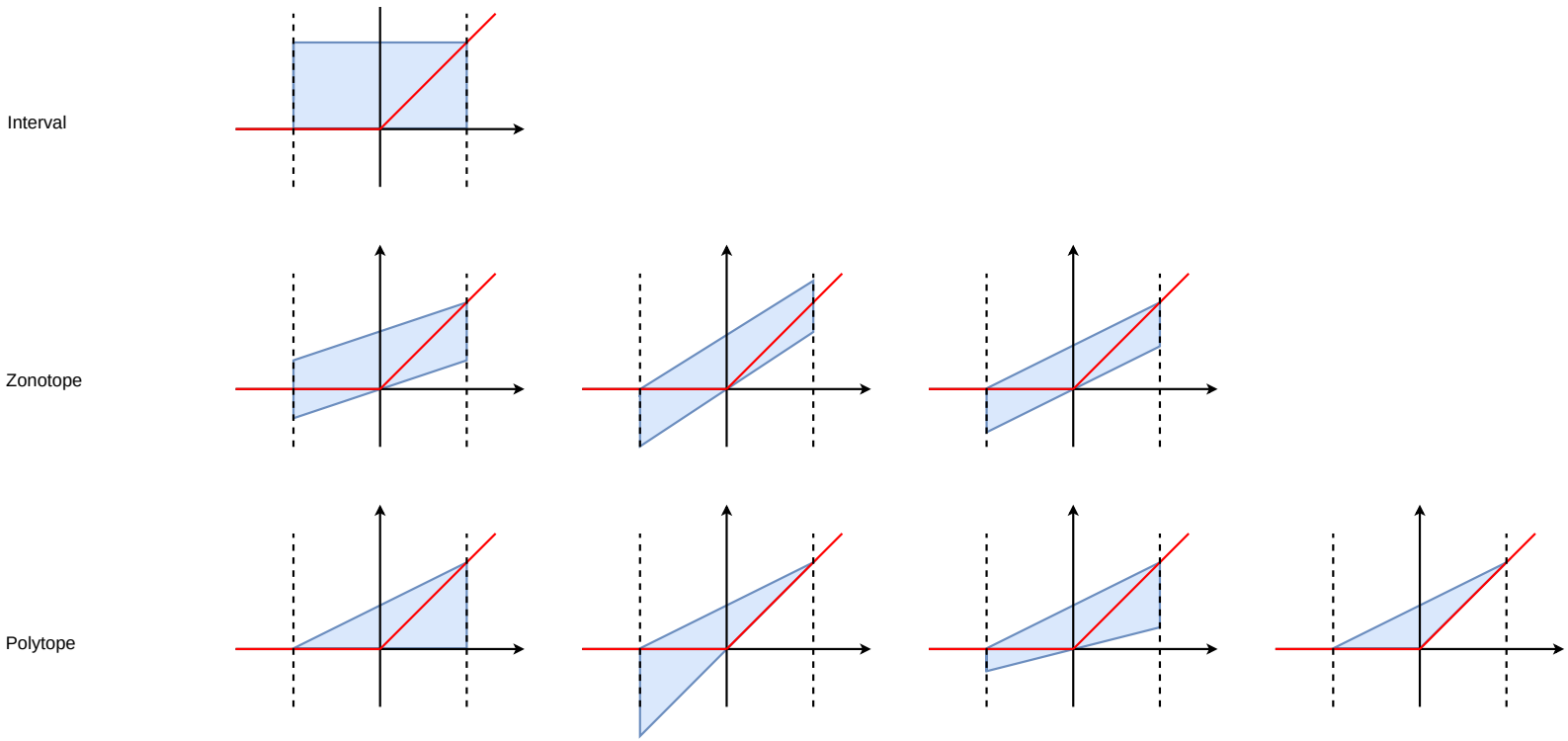
Answer: Yes / No



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

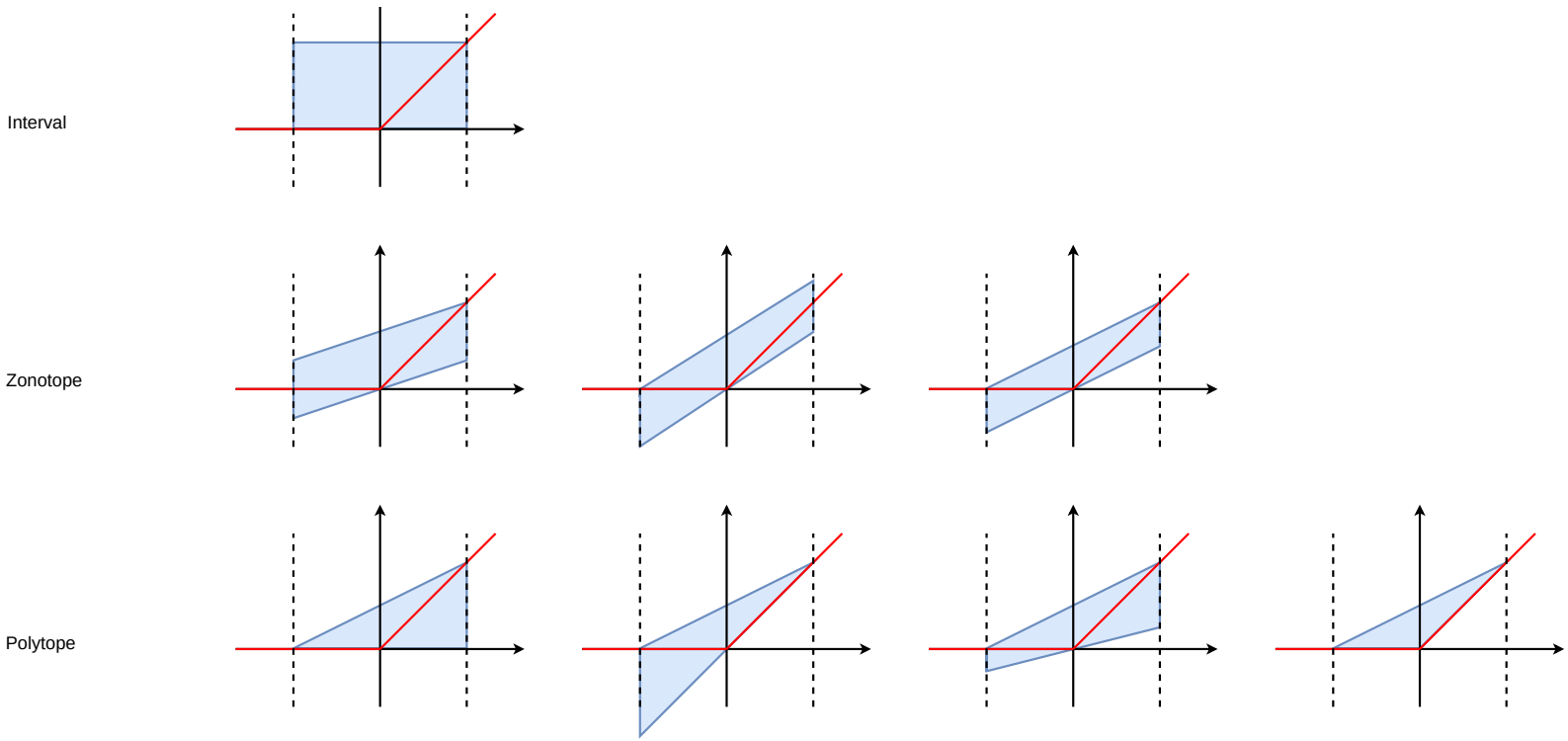
Abstraction

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval (ReluVal), zonotopes (ERAN), polytopes (α, β-CROWN)



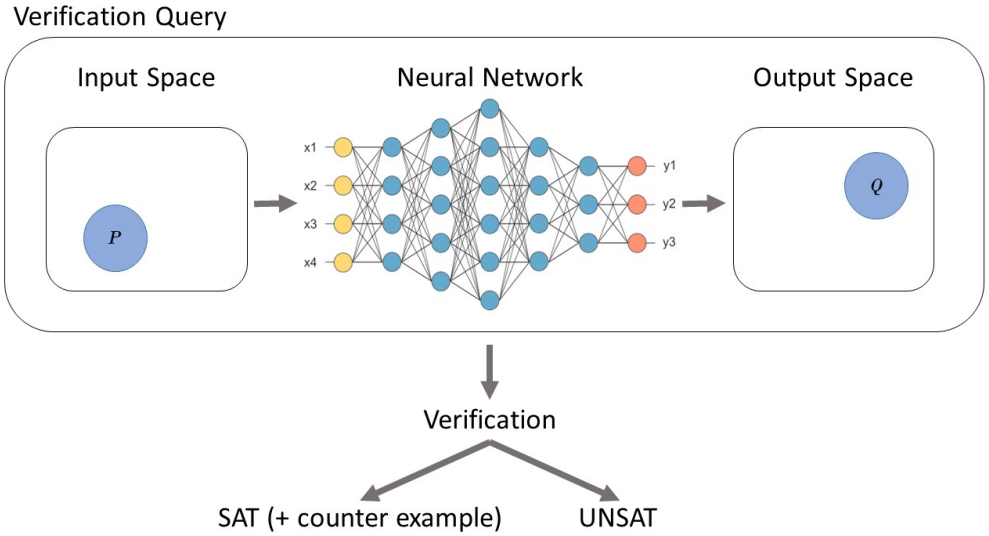
Abstraction

- Overapproximate computation (e.g., ReLU) using abstract domains
  - interval (ReluVal), zonotopes (ERAN), polytopes (α, β-CROWN)

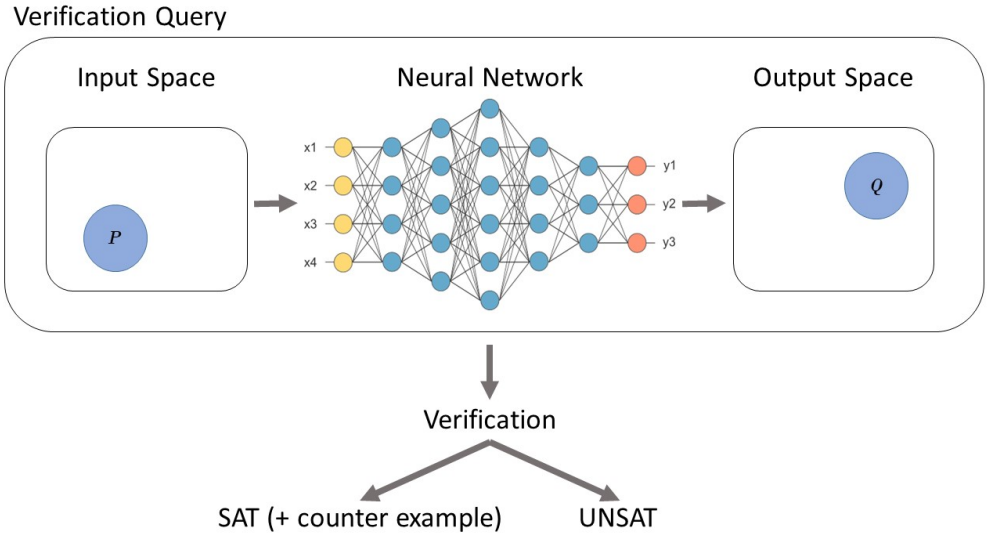


- Scale well, but *loose precision* (producing spurious cex's)
  - Newer work: iterative refine abstraction to filter spurious cex's

Constraint Solving

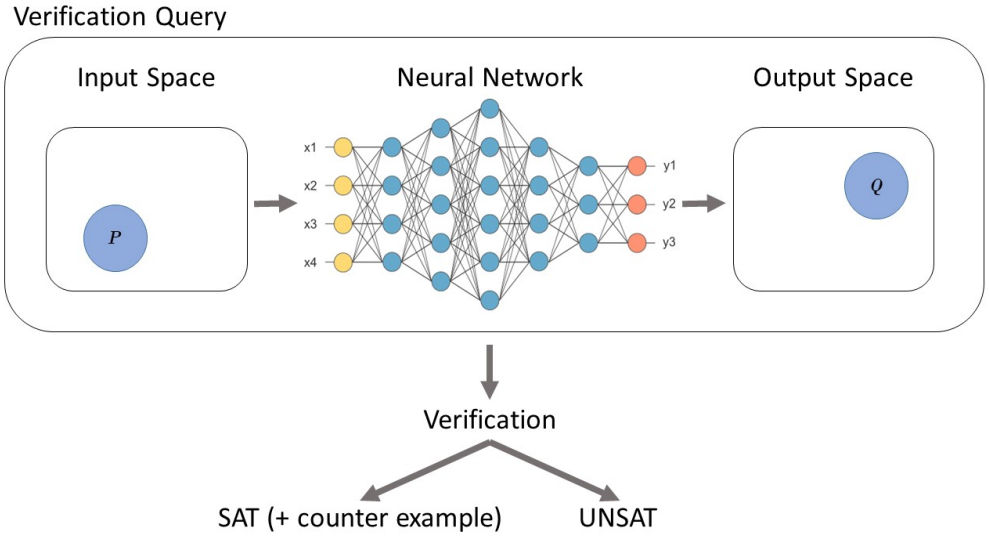


Constraint Solving



- 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., $N \wedge P \wedge \neg Q$ is satisfiable
    - UNSAT: p is a property of N
    - SAT: p is not a property of N (also give counterexample inputs satisfiying P but not Q)

Constraint Solving



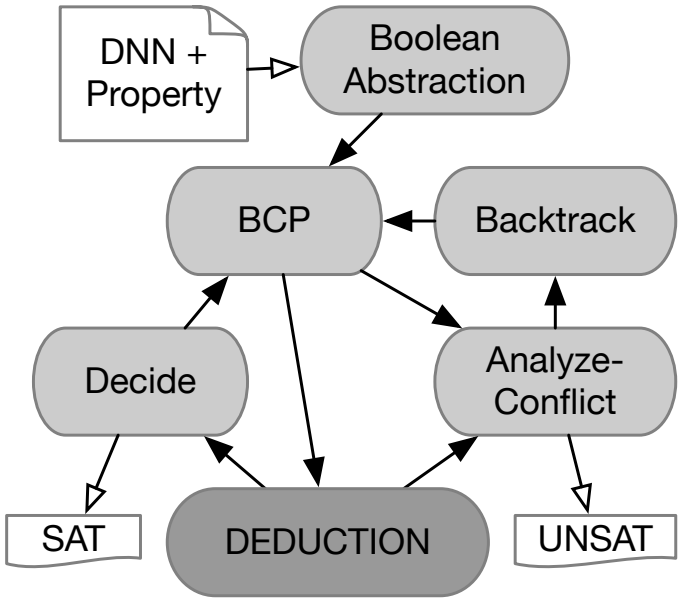
- Transform DNN verification into a constraint (satisfiability) problem
  - To prove $N \Rightarrow p$ (where p is $P \Rightarrow Q$)
    - check if $\neg(N \Rightarrow (P \Rightarrow Q))$, i.e., $N \wedge P \wedge \neg Q$ is satisfiable
    - UNSAT: p is a property of N
    - SAT: p is not a property of N (also give counterexample inputs satisfiying 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

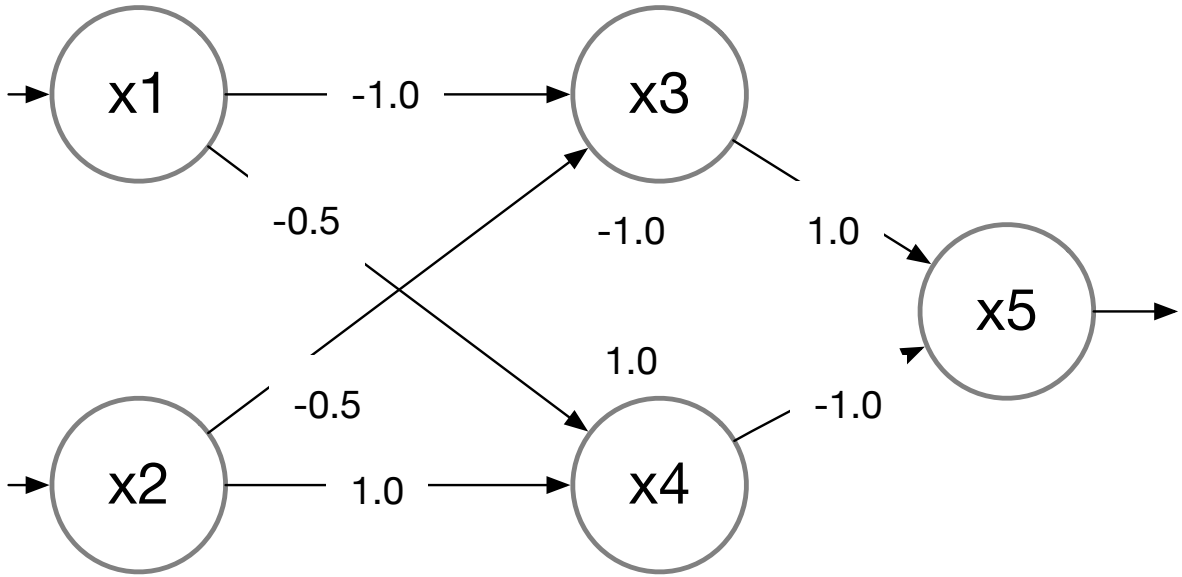


Use NeuralSAT to prove $N \Rightarrow (P \Rightarrow Q)$

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

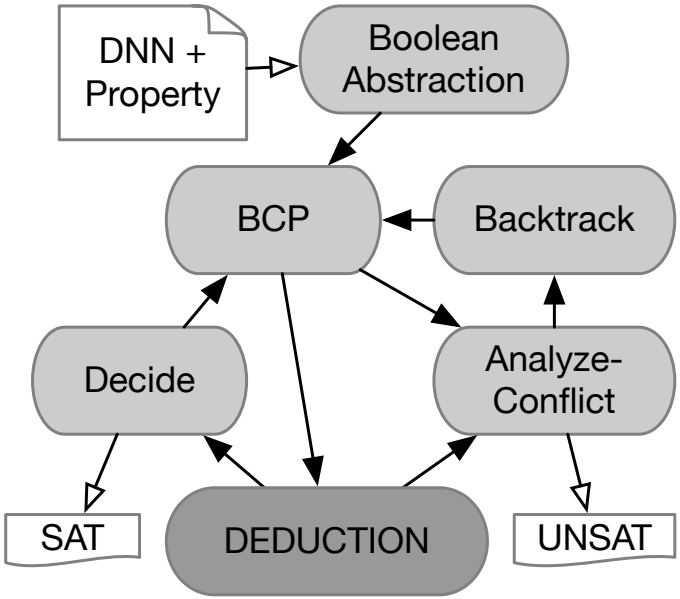
Insight: combines conflict clause learning in SAT solving and abstraction for scalability

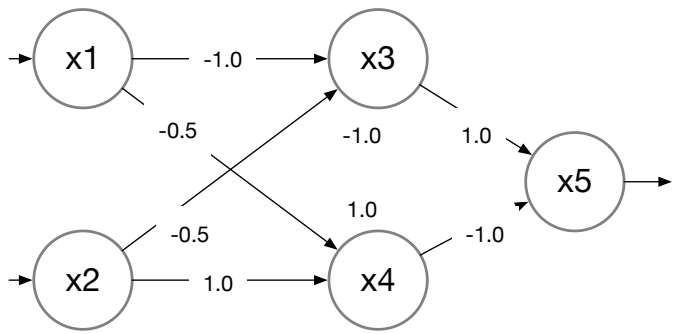
Example



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

- NeuralSAT($\neg f$) =
  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

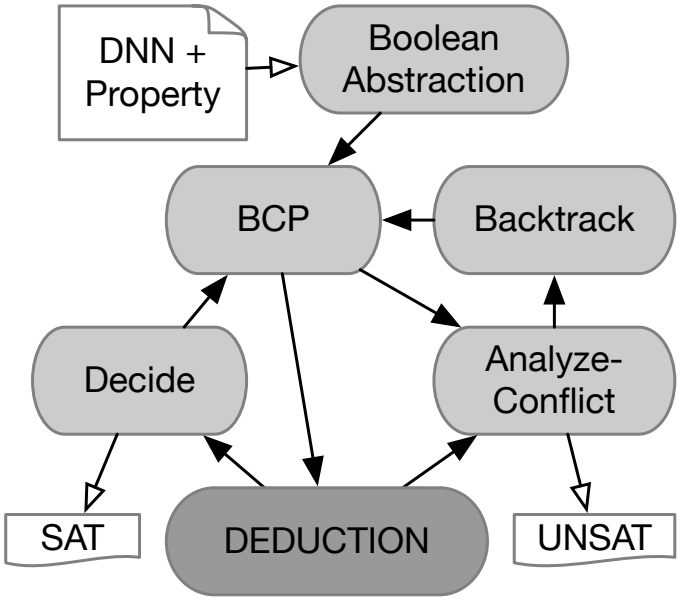


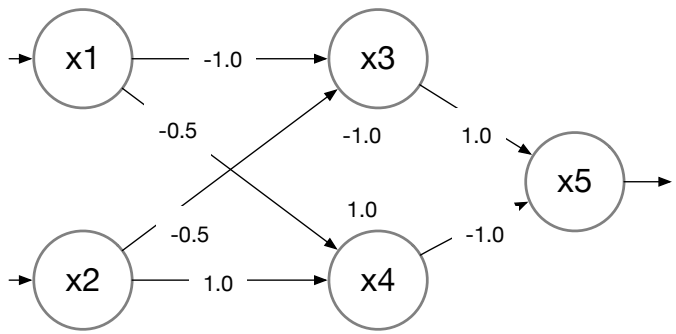


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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, $-x_1 - 0.5x_2 - 1 > 0$
- Form two clauses $\{v_3 \vee \overline{v_3}\ ;\ v_4 \vee \overline{v_4}\}$
- Find boolean values for v_3, v_4 that satifies the clauses and their implications

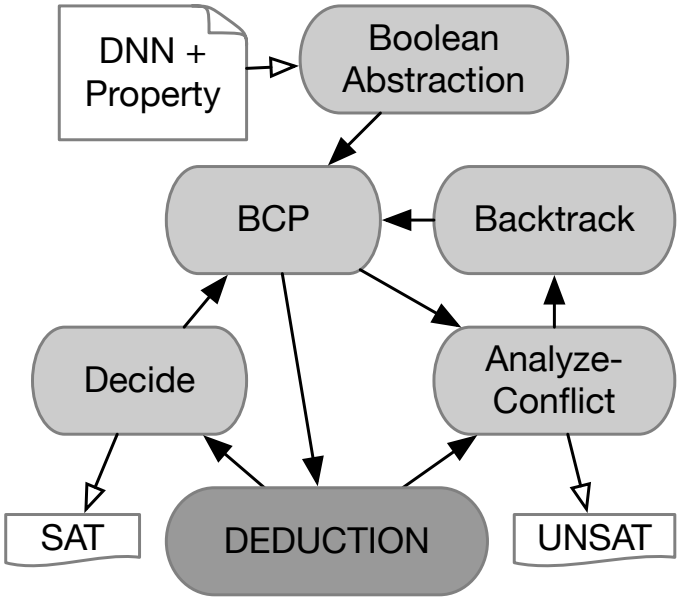


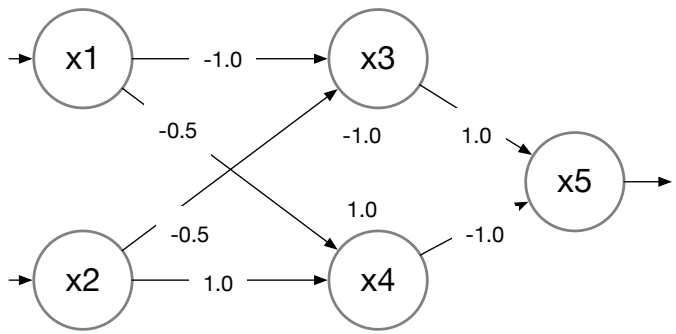


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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 $-x_1 - 0.5x_2 - 1 < 0$

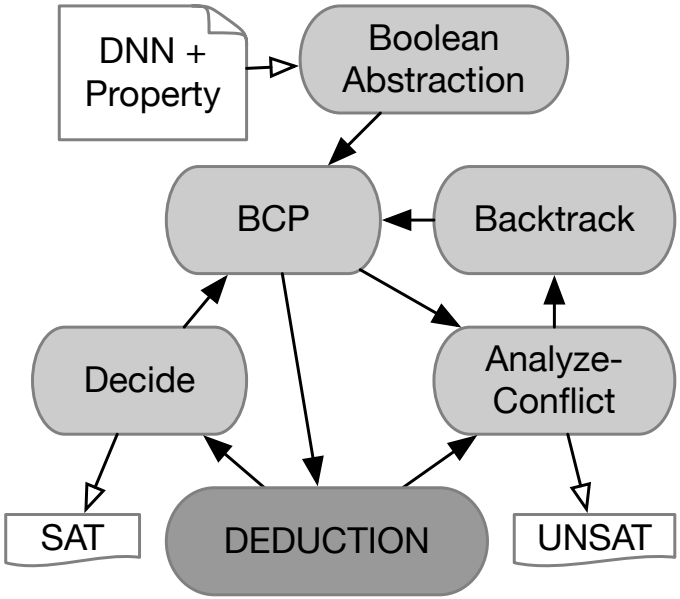


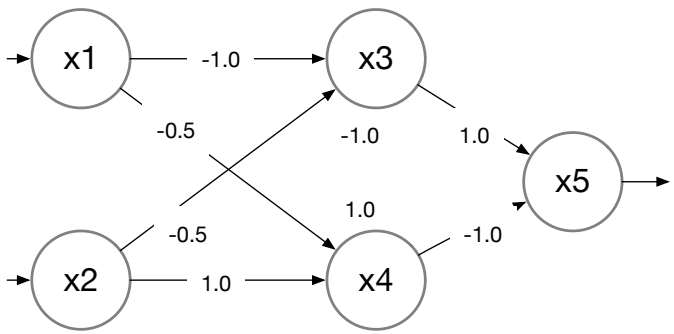


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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

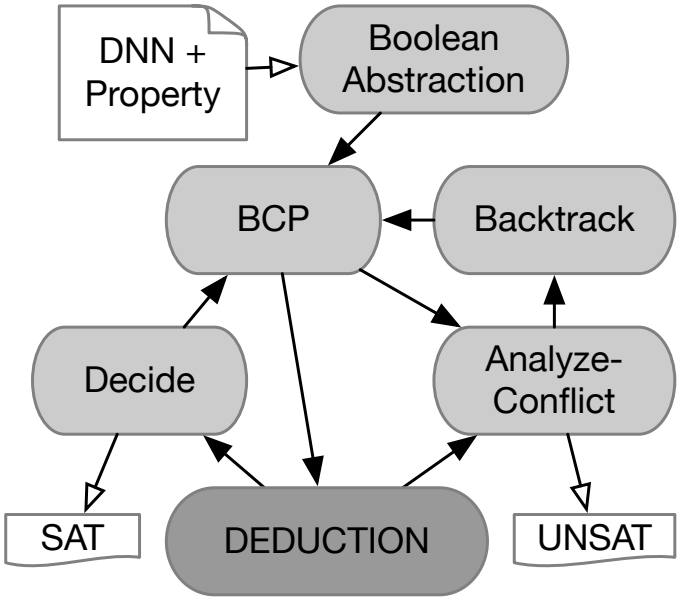


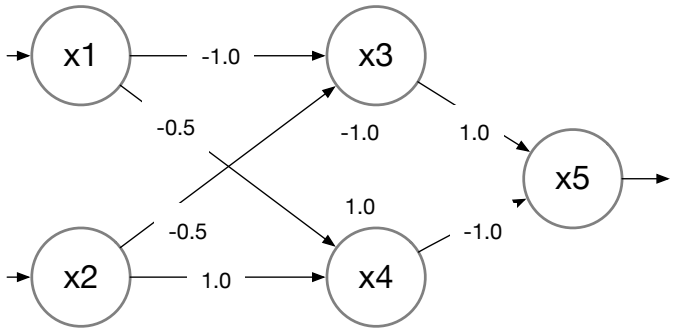


$x_1 \in [-1, 1], x_2 \in [-2, 2], x_5 > 0$

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)
- $\vdots$





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

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
- **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)

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**

Prop	**NeuralSAT**	**Marabou**
ϕ_1	1025.36	TO (3 hrs)
ϕ_2^*	**22.84**	821.41
ϕ_3	**526.77**	8309.09
ϕ_4	330.83	**133.97**
ϕ_5	**83.51**	1259.74
ϕ_6	**127.35**	250.41
ϕ_7^*	**262.01**	TO
ϕ_8^*	**0.15**	TO
ϕ_9	**142.00**	TO
ϕ_{10}	**191.99**	3134.35

ACAS XU Results

Much faster than the constraint solver **Marabou**

Prop	NeuralSAT	Marabou
ϕ_1	1025.36	TO (3 hrs)
ϕ_2^*	**22.84**	821.41
ϕ_3	**526.77**	8309.09
ϕ_4	330.83	**133.97**
ϕ_5	**83.51**	1259.74
ϕ_6	**127.35**	250.41
ϕ_7^*	**262.01**	TO
ϕ_8^*	**0.15**	TO
ϕ_9	**142.00**	TO
ϕ_{10}	**191.99**	3134.35

- 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