UNSAT Configuration을 활용한 효율적 심층 신경망 검증

채승현

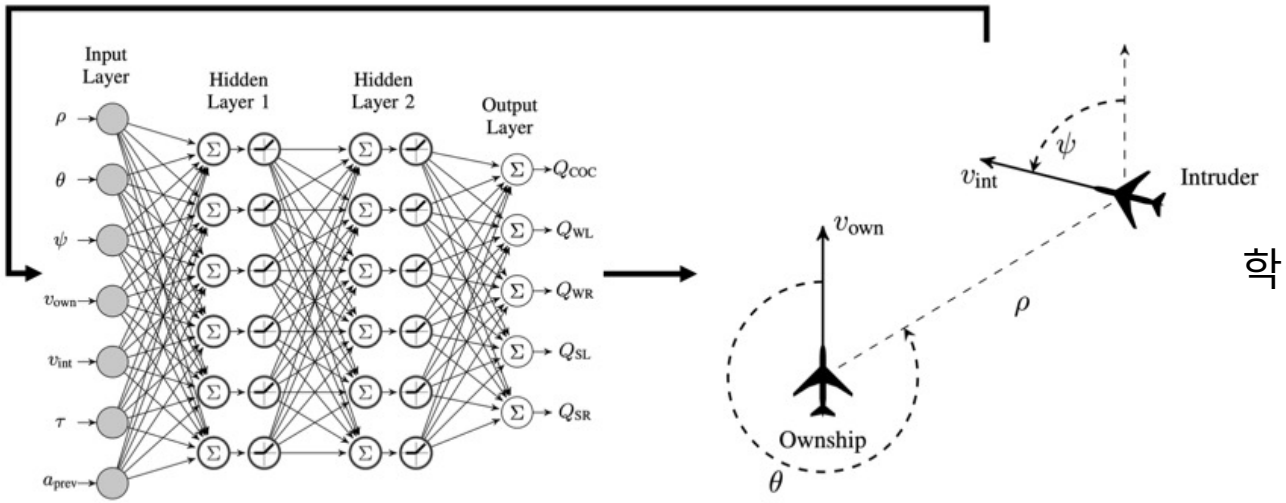
2024/7/9

POSTECH Software Verification Lab

Part 1

해결하고자 하는 문제 및 연구 핵심 아이디어

심층 신경망 검증 문제



학습된 심층 신경망

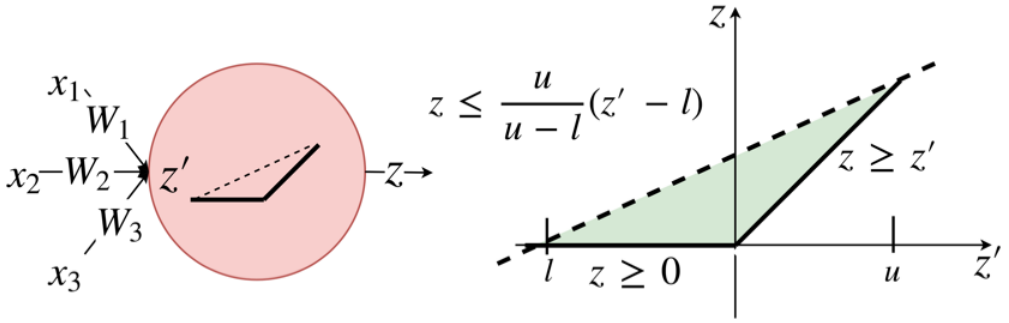
검증하고자 하는 요구사항/성질

Property ϕ_2.

- Description: If the intruder is distant and is significantly slower than the ownship, the score of a COC advisory will never be maximal.
- Tested on: $N_{x,y}$ for all $x \geq 2$ and for all y.
- Input constraints: $\rho \geq 55947.691$, $v_{\text{own}} \geq 1145$, $v_{\text{int}} \leq 60$.
- Desired output property: the score for COC is not the maximal score.

해결하고자 하는 문제

- SOTA: Symbolic Bound Propagation with Iterative Hidden Node Refinement

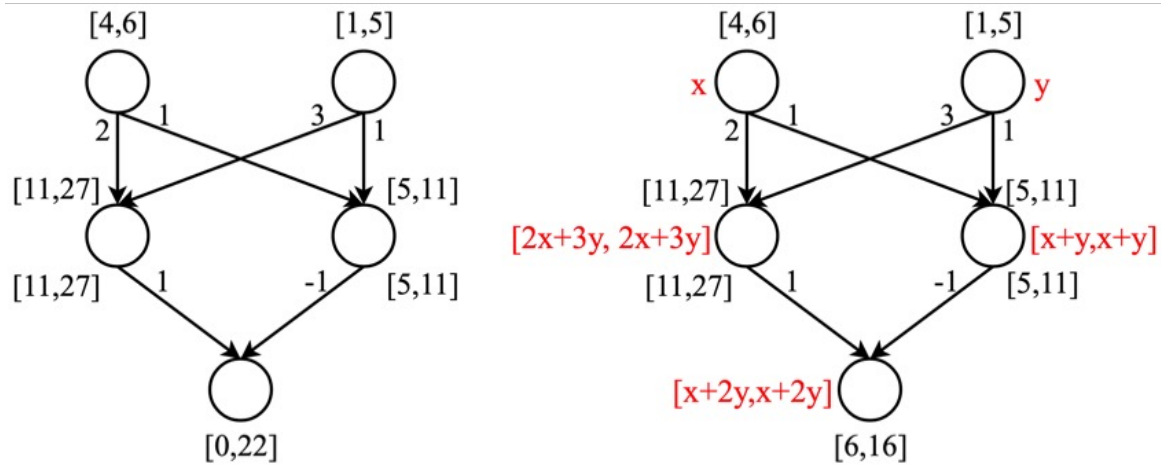


(bound propagation을 할 수 있도록)

Non-linear한 ReLU 활성화 함수를 linear하도록 abstract

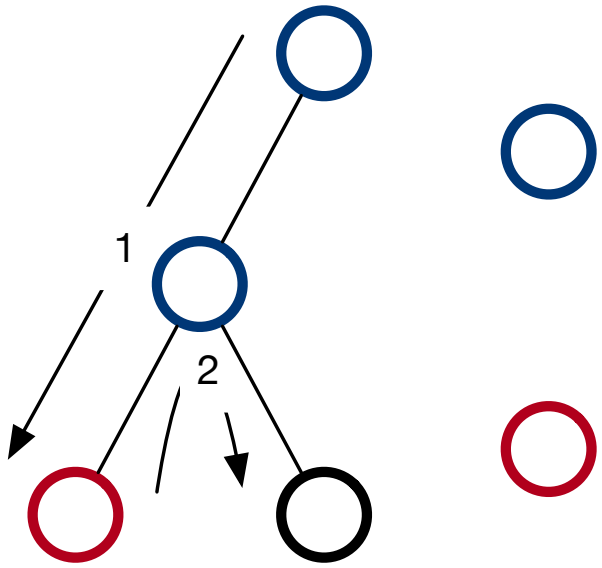
Hidden 및 output node가 가질 수 있는 값의 범위를

(symbolic 하게) 입력 변수로 표현



해결하고자 하는 문제

- SOTA: Symbolic Bound Propagation with Iterative Hidden Node Refinement



Satisfactory Configuration

- There might exist a counterexample input that dissatisfies the desirable property.
- If found false positive, split another abstract ReLU.

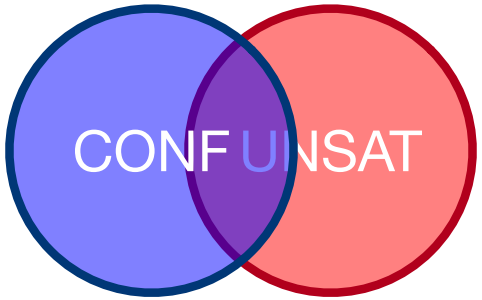
Unsatisfactory Configuration

- There is no input that dissatisfies the given property, given current ReLU statuses.

➔ 현재 configuration이 UNSAT임이 확인된 이후, 바로 다음 conf으로 넘어감

핵심 아이디어

- UNSAT conf ψ_u 가 가지고 있는 정보
  - 아직 SAT인지, UNSAT인지 모르는 conf ψ_c 가 있을 때,
  - Conf ψ_c와 UNSAT conf ψ_u가 intersect하는 부분은 별도의 계산 없이 안전함을 알 수 있다.



$$\psi_c(\vec{v}) \setminus \psi_u(\vec{v}) := \psi_c(\vec{v}) \wedge \neg\psi_u(\vec{v})$$

이제 확인해야 하는 conf ψ_c'

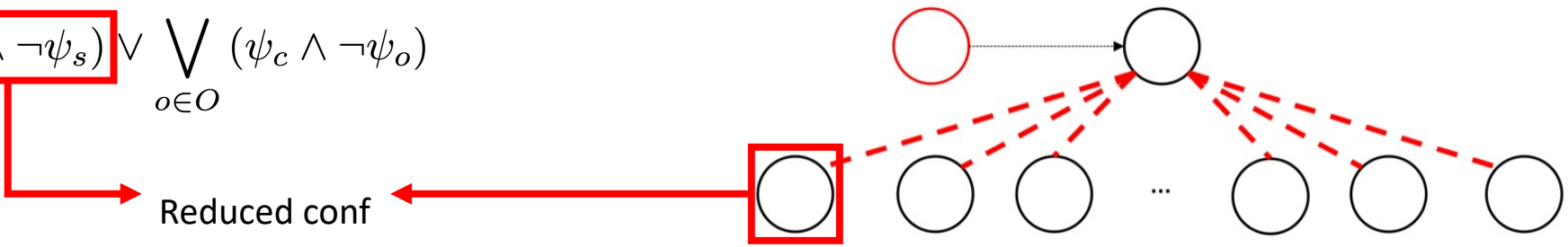
핵심 아이디어

$$\psi_c \wedge \neg\psi_u = \psi_c \wedge \neg\left(\bigwedge_{i\in I}\psi_i \wedge \bigwedge_{s\in S}\psi_s \wedge \bigwedge_{o\in O}\psi_o \wedge \neg property\right)$$

$$= \psi_c \wedge \left(\bigvee_{i\in I}\neg\psi_i \vee \bigvee_{s\in S}\neg\psi_s \vee \bigvee_{o\in O}\neg\psi_o \vee property\right)$$

$$= \bigvee_{i\in I}(\psi_c \wedge \neg\psi_i) \vee \bigvee_{s\in S}(\psi_c \wedge \neg\psi_s) \vee \bigvee_{o\in O}(\psi_c \wedge \neg\psi_o) \vee (\psi_c \wedge property)$$

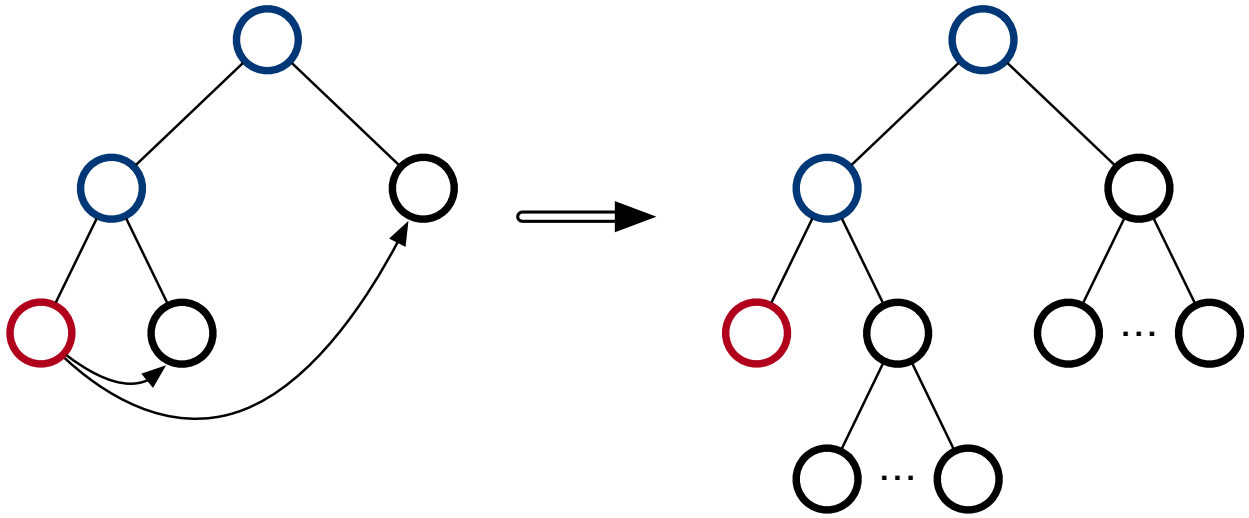
$$= \bigvee_{s\in S}(\psi_c \wedge \neg\psi_s) \vee \bigvee_{o\in O}(\psi_c \wedge \neg\psi_o)$$



➔ 즉, UNSAT conf를 '적용'함으로써, 여러 개의 reduced conf가 생성

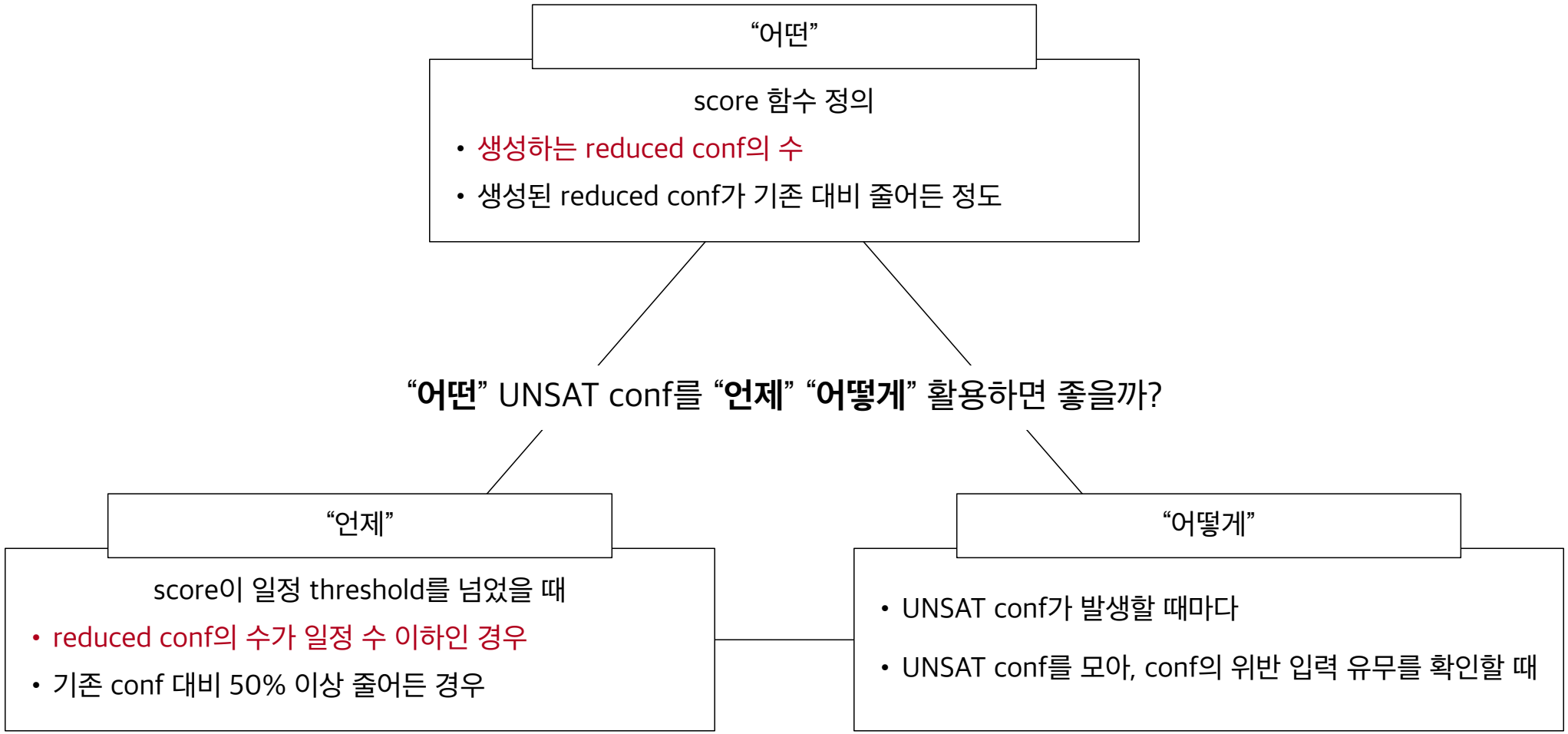
핵심 아이디어 문제

- 가장 naïve한 방식: UNSAT conf가 발생할 때마다, 모든 conf에 적용



➔ 확인이 필요한 input region 감소 효과 <<< 기하급수적으로 증가하는 reduced conf 수

연구 핵심 포인트


“어떤”
score 함수 정의
• 생성하는 reduced conf의 수
• 생성된 reduced conf가 기존 대비 줄어든 정도
“**어떤**” UNSAT conf를 “**언제**” “**어떻게**” 활용하면 좋을까?
“언제”
score이 일정 threshold를 넘었을 때
• reduced conf의 수가 일정 수 이하인 경우
• 기존 conf 대비 50% 이상 줄어든 경우
“어떻게”
• UNSAT conf가 발생할 때마다
• UNSAT conf를 모아, conf의 위반 입력 유무를 확인할 때

Part 2

기존 연구 및 현재 진행 중인 연구

Main Idea

- 적용했을 때, reduced conf를 하나만 생성하는 UNSAT conf들이 존재함을 확인
  - $|Impl| = m - 1$ 인 경우 (score = 1)

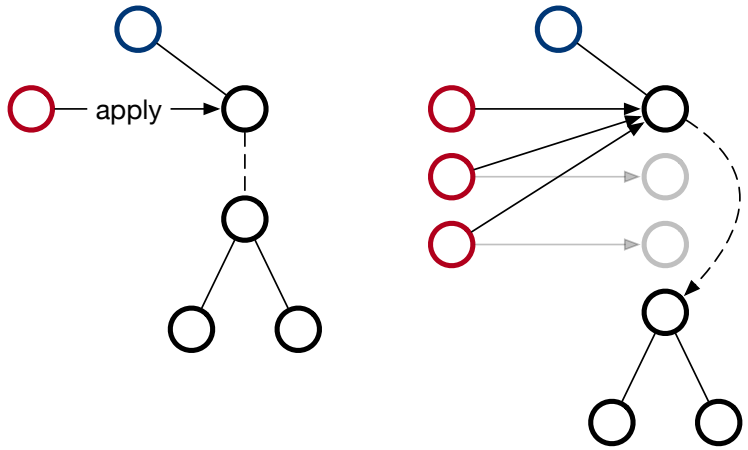
$$\text{Given } \psi_c = \bigwedge_n \psi_c^n,\ \psi_u = \bigwedge_m \psi_u^m$$

$$\psi_c \wedge \neg\psi_u \equiv \psi_c \wedge \neg \left(\bigwedge_{(*,j) \notin Impl_{c,u}} \psi_u^j \right) \text{ where}$$

$$\texttt{implication set } Impl_{c,u} := \{(i,j) \mid \psi_c^i \implies \psi_u^j \text{ and } i \leq n \text{ and } j \leq m\}$$

Main Idea

- score = 1 UNSAT conf를 한 개 적용했을 때 문제점
  - 추가된 constraint 하나로 인해 배제되는 computation의 수가 많지 않다
  - score을 계산하는 비용 대비 computation 감소 효과 미비



$$\psi_c \bigwedge_{u \in ucf} \neg\psi_u \equiv \psi_c \bigwedge_{u \in ucf} \neg \left(\bigwedge_{(*,j) \notin Impl_{c,u}} \psi_u^j \right)$$

➔ 보관해둔 모든 UNSAT conf 중에 score = 1 UNSAT conf들을 추출해 동시에 적용

기존 연구/방식의 한계

Property	Conf Tree Size		Verification Time (seconds)	
	No Unsat	Naïve Use	No Unsat	Naïve Use
Prop 2 (tiny2)	311.04	274.78	0.43	2.99
Prop 2 (tiny3)	5919.1	4531.05	15.45	384.73

- UNSAT conf의 정보가 유의미함과 활용 가능함을 확인했으나,
- 활용 비용이 너무 큼

현재 연구의 핵심

- 간단한 휴리스틱들을 통해
  - 검증 과정에서 활용할 가치가 있는지 판단하고, 저장할 UNSAT conf를 필터링
  - 각 conf는 자기 자신에 가장 적용될 확률이 높은 UNSAT conf들만 모티너링