Utilization of Activation Property DB for Efficient Incremental Neural Network Verification

채승현, 연주은, 배경민

POSTECH Software Verification Lab

2025년 STAAR 여름 정기 워크샵

Requirements for Neural Network

- Safety: 주어진 입력 범위에 대해 신경망은 원하는 범위 내 값을 출력해야 한다.

  예: 각 입력 노드의 값이 10 이하일 경우, 첫 번째 출력 노드의 값은 0 이하여야 함

- Local robustness: 특정 입력 값에 대해 신경망은 일정 범위 내 변화에 대해 똑같이 출력해야 한다.

  예: 고양이, 강아지 사진을 구분하는 신경망은, 고양이 사진에 미세한 수정이 가해지더라도 올바르게 판별해야 함

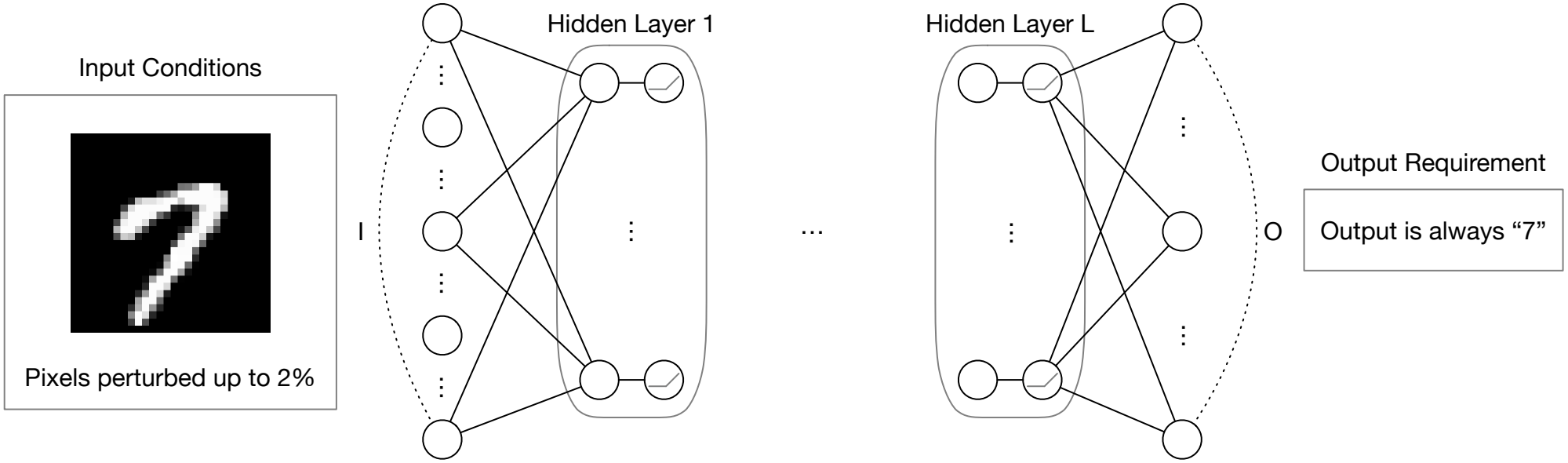
- Fairness: 출력이 사용된 학습 데이터에 내제된 편향을 그대로 반영하면 안 된다.

  예: 사용자의 신용도를 구분하는 신경망은, 사용자의 나이에 대해 공정해야 함

Neural Network Verification

- 주어진 입력 범위와 출력 조건을 위반하는 입력이 존재하는지 확인하는 기술

  → 검증 문제: 신경망 + 요구사항 (입력 범위, 출력 조건)

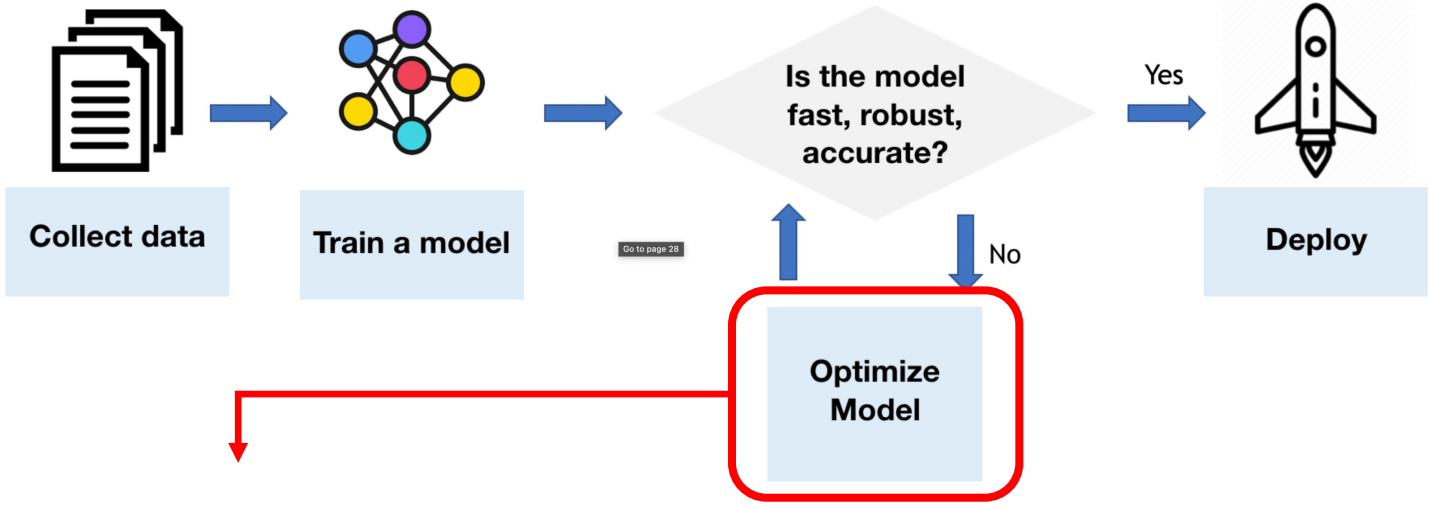


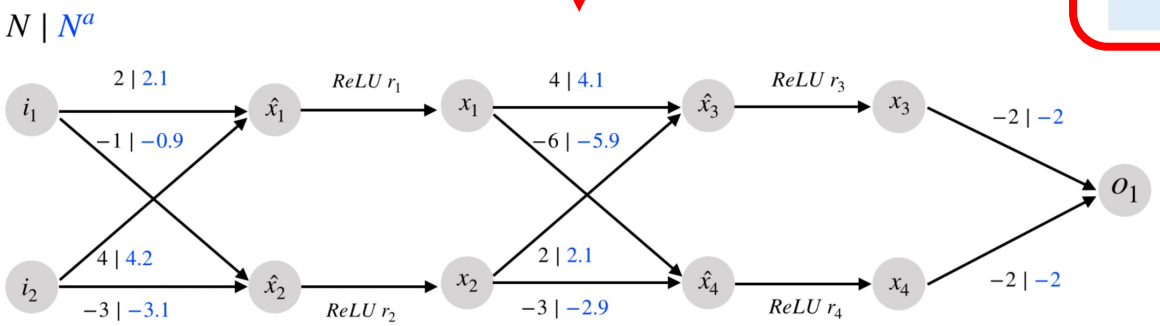
신경망이 사용자가 의도하는 대로 행동하는지 확인하는 매우 중요한 과정

Digit Image Source: MNIST (LeCun et al.)

Continuous Optimization of Neural Networks for Deployment

- 보통 학습된 모델은 곧바로 사용되지 않으며, 성능 향상을 위한 최적화 과정을 거치게 됨





기존 신경망과 “유사한” 신경망이 그대로 요구사항을 만족하는지, 재검증이 필요

→ 매우 비효율적!

Source: Ugare et al., “Proof Transfer for Fast Certification of Multiple Approximate Neural Networks”, OOPSLA 2022
Source: Ugare et al., “Incremental Verification of Neural Networks”, PLDI 2023

What is Incremental Neural Network Verification?

이전 검증 과정의 정보를 활용하여,

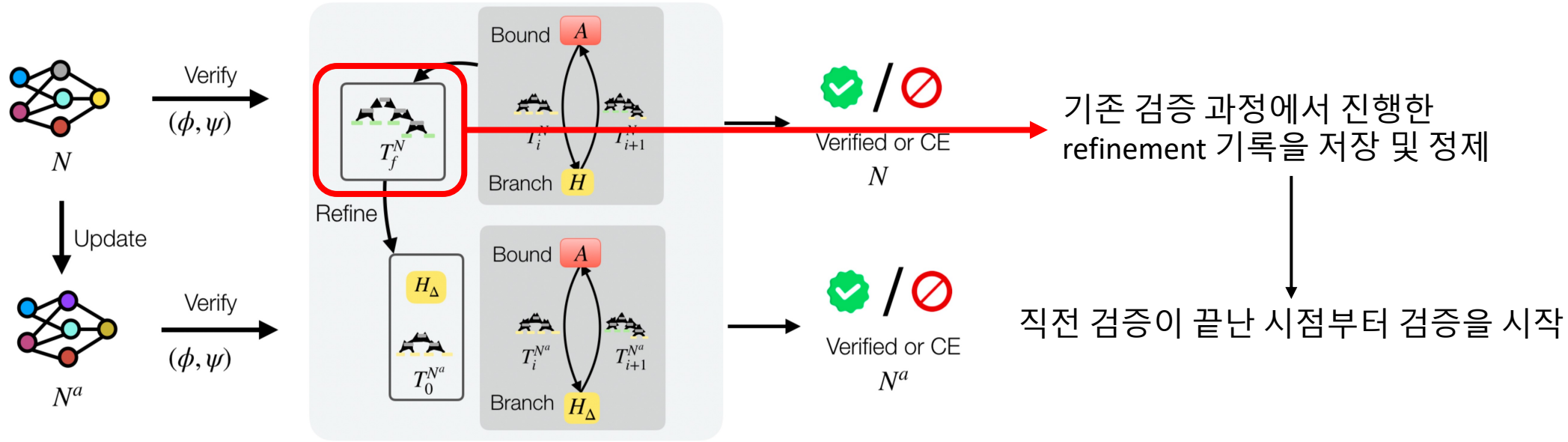
최적화 과정을 통해 변한 신경망의 검증을

처음부터 다시 수행하지 않아도 되도록 하여

검증 성능을 향상시키는 기술

Previous Works on Incremental NN Verification

- IVAN[PLDI 2023], Olive[OOPSLA1 2024]
  - 대상: Abstraction-and-Refinement와 Branch-and-Bound(BaB) 기법을 활용하는 검증 기법



Workflow of IVAN. Source: Ugare et al., "Incremental Verification of Neural Networks", PLDI 2023

Limitations of Previous Works

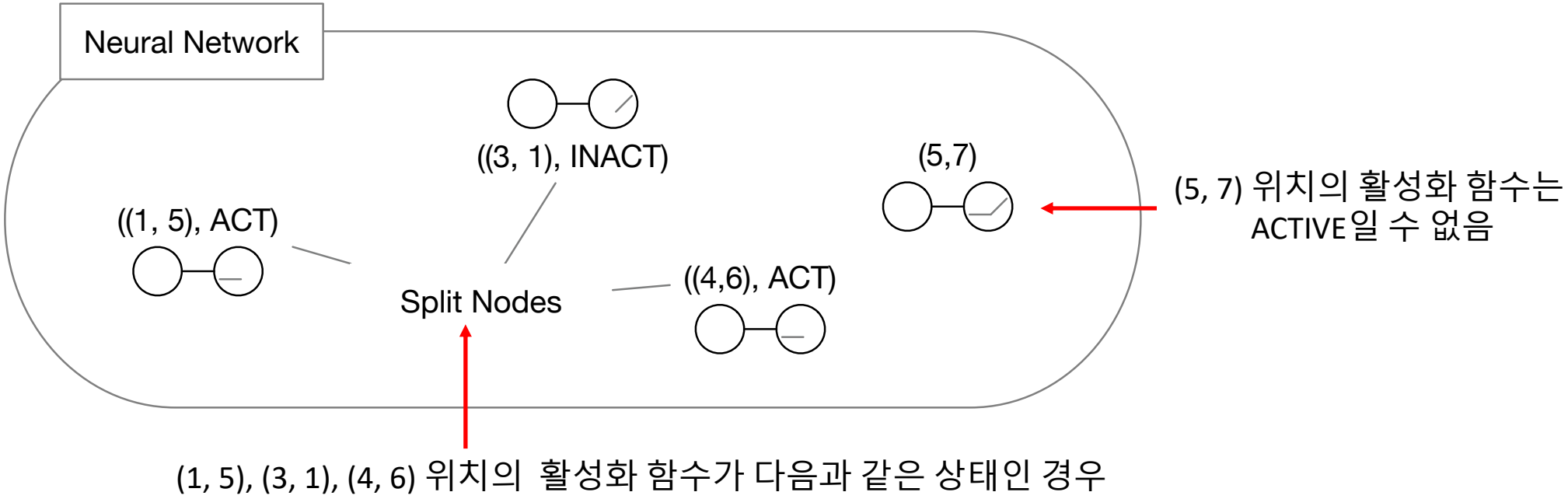
- IVAN과 Olive의 한계점
  - BaB 기법에만 제한적으로 적용 가능하며,
  - 휴리스틱한 이전 정보 활용으로 인해 검증 성능을 확실하게 보장하지 못하며, 확률적으로 효과를 발휘

기존 Incremental Verification 연구의 한계점을 극복할 수 있는
”Activation Property”라는 정보 제안 및 이를 활용하는 연구 진행

What is an “Activation Property”

- Activation property: 특정 활성화 함수들의 상태가 다른 특정 활성화 함수들의 상태를 imply하는 성질

Activation Property: ((1, 5), ACT) ∧ ((3, 1), INACT) ∧ ((4, 6), ACT) → not ((5, 7), ACT)

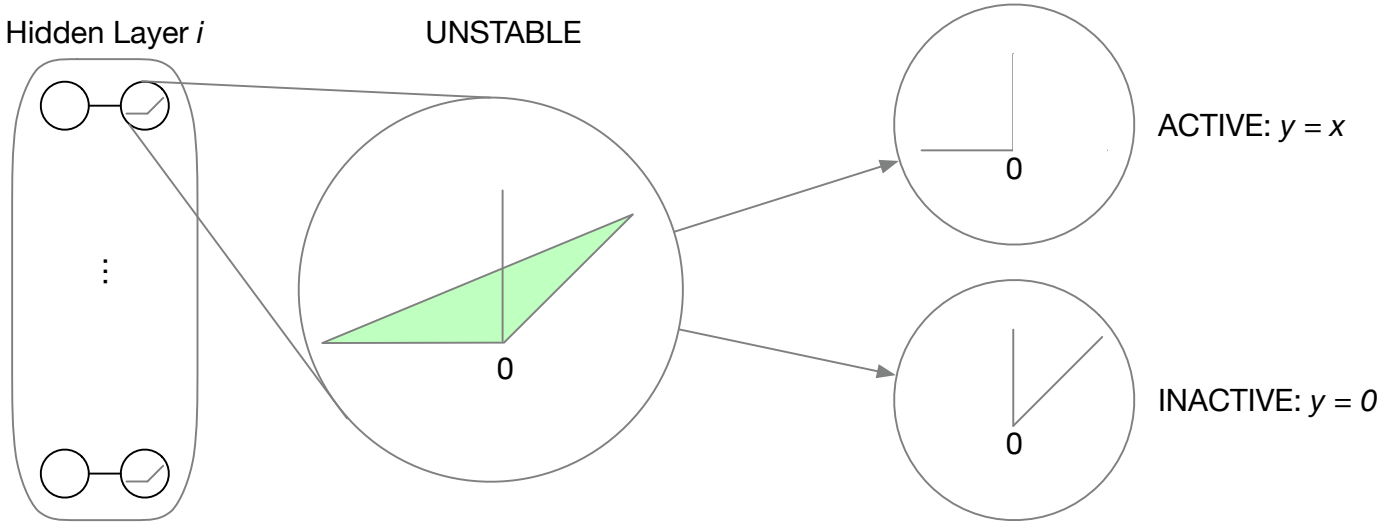


Applicable Technique: Abstraction-and-Refinement & BaB

- Abstraction-and-Refinement & BaB

  → 신경망을 구성하는 모든 활성화 함수를 추상화한 후, 이를 하나씩 refine해 나가며 (“split” for ReLU)

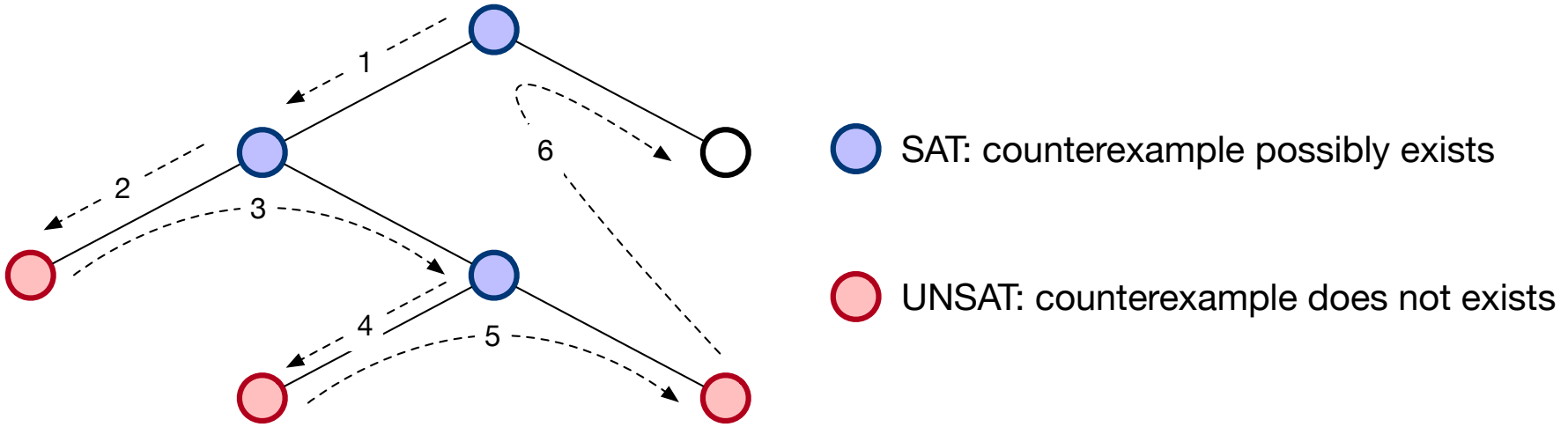
  → divide-and-conquer와 비슷하게 tree 구조를 따라 문제를 분할하며 해결하는 접근 방식



Applicable Technique: Abstraction-and-Refinement & BaB

- 생성되는 문제들의 tree를 배회/탐색하며, 검증하고자 하는 요구사항을 위반하는 입력이 존재하는지 확인

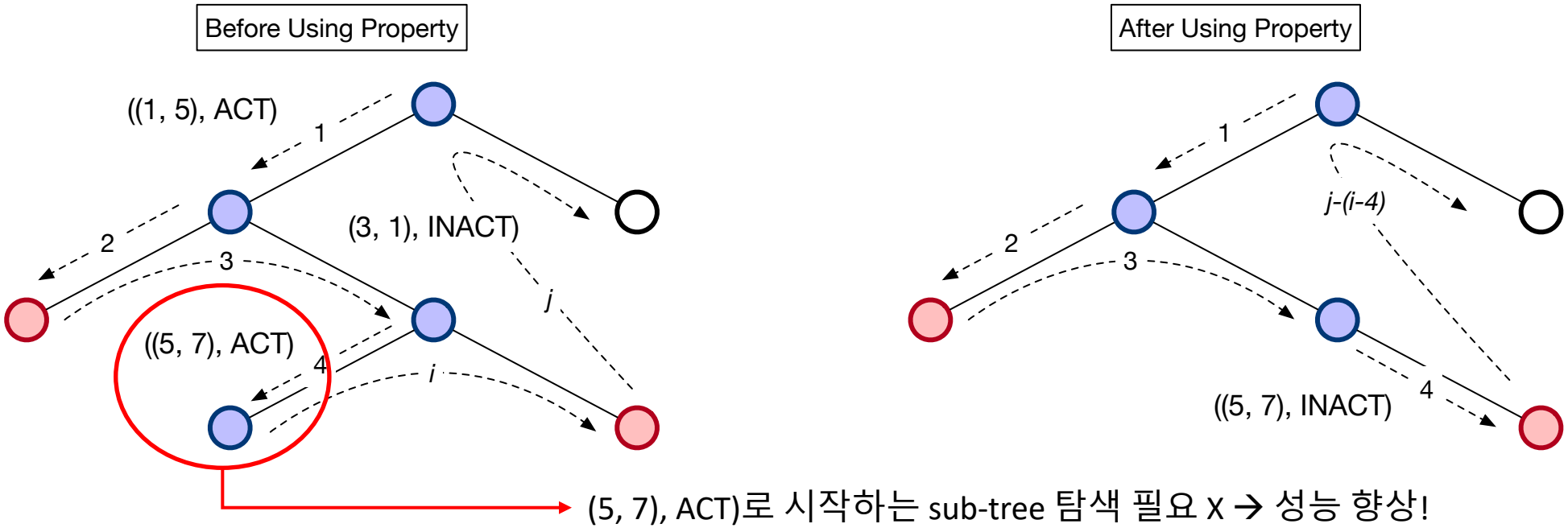
  → 모든 leaf 문제들에 대해 위반입력을 찾지 못하면 요구사항이 만족



탐색해야 하는 tree의 크기가 검증 성능을 좌우

Usefulness of Activation Property

Activation Property: ((1, 5), ACT) ∧ ((3, 1), INACT) → not ((5, 7), ACT)



(5, 7), ACT)로 시작하는 sub-tree 탐색 필요 X → 성능 향상!

기존에 존재하는 탐색해야 하는 tree의 크기를 줄이는 정보 및 활용 방식을 Activation Property로 표현 가능

Contribution: Activation Property DB Utilizing Framework

1. 이전 검증 과정으로부터 Activation Property 추출

   → 검증 기법의 핵심 요소인 활성화 함수 상태에 대한 유용한 성질 추출

2. 추출된 성질로 데이터베이스 구축

   → 이후 다양한 검증 과정에서 재사용 가능

3. 이후 검증 과정에서 데이터베이스 활용

   → Activation Property를 통해 신경망 검증 성능 향상

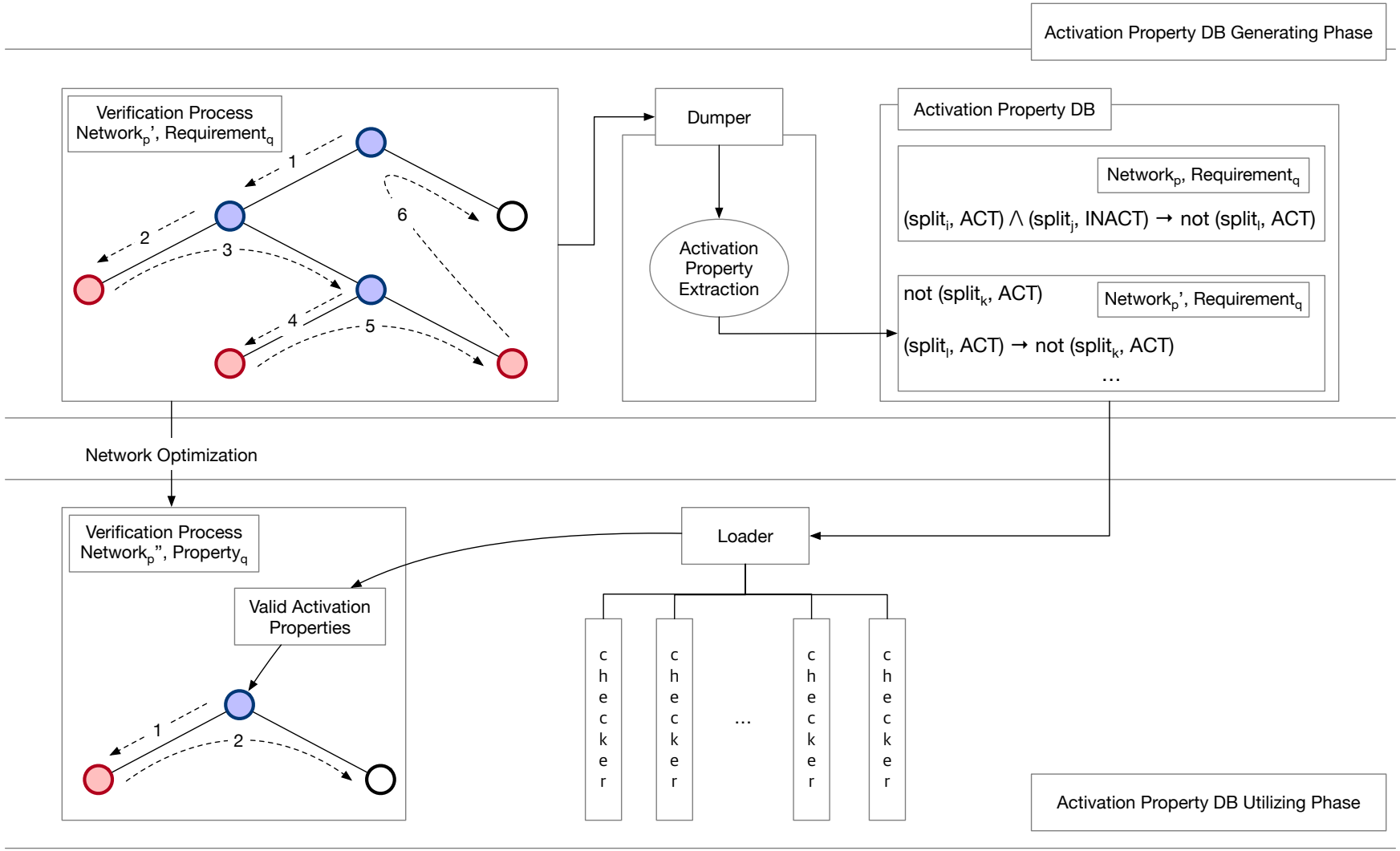
Contribution: Activation Property DB Utilizing Framework

이전 검증 과정의 정보를 활용하는

기존 검증 기법들을 포괄하고,

확장한 프레임워크를 제안 및 구현

Activation Property DB Utilizing Framework Overview

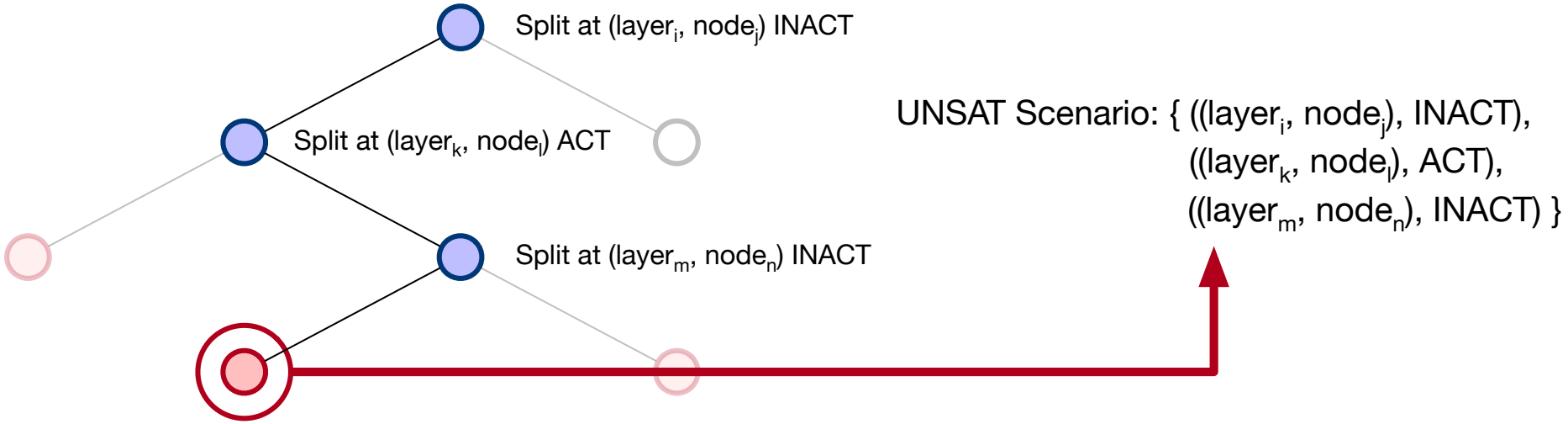


DB Generating Phase: Extracting Activation Property

- UNSAT Scenario를 계산하여 추출

  → UNSAT Scenario: UNSAT 결과를 유발한 split들의 집합 (해당 split들이 진행된 경우, 위반 입력이 존재하지 않는다)

  → 일종의 Activation Property

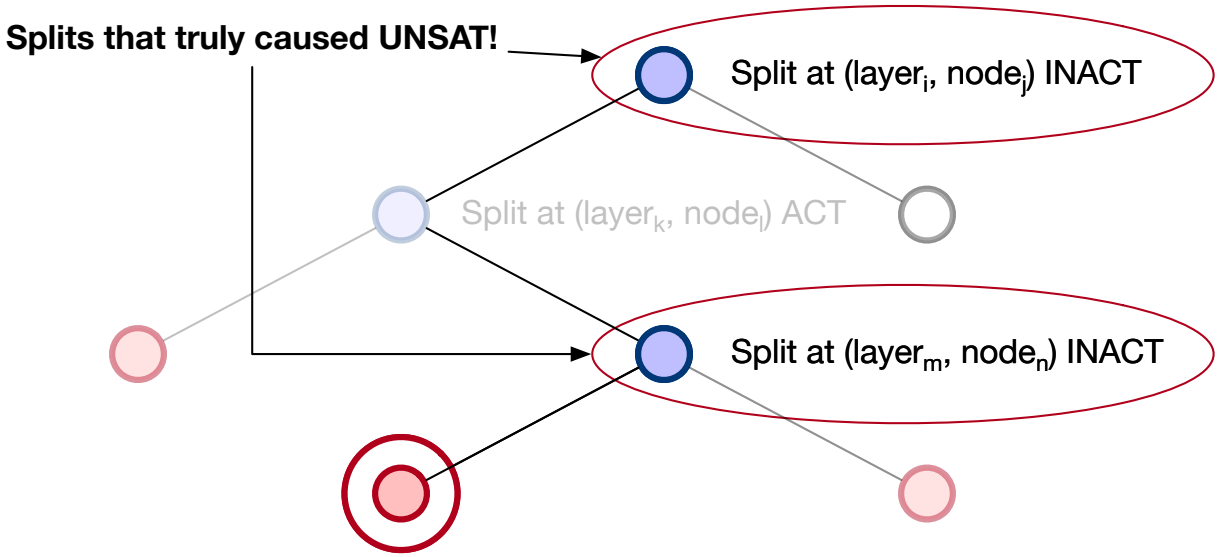


Activation Property: ((layer$_i$, node$_j$), INACT) /\ ((layer$_k$, node$_l$), ACT) → not ((layer$_m$, node$_n$), INACT)

DB Generating Phase: Extracting Activation Property

- UNSAT Scenario를 generalization함으로써 Activation Property 강화

  → Generalization: 실제로 UNSAT 결과를 유발한 split만 추출



UNSAT Scenario: { ((layer$_i$, node$_j$), INACT),
((layer$_k$, node$_l$), ACT),
((layer$_m$, node$_n$), INACT) }

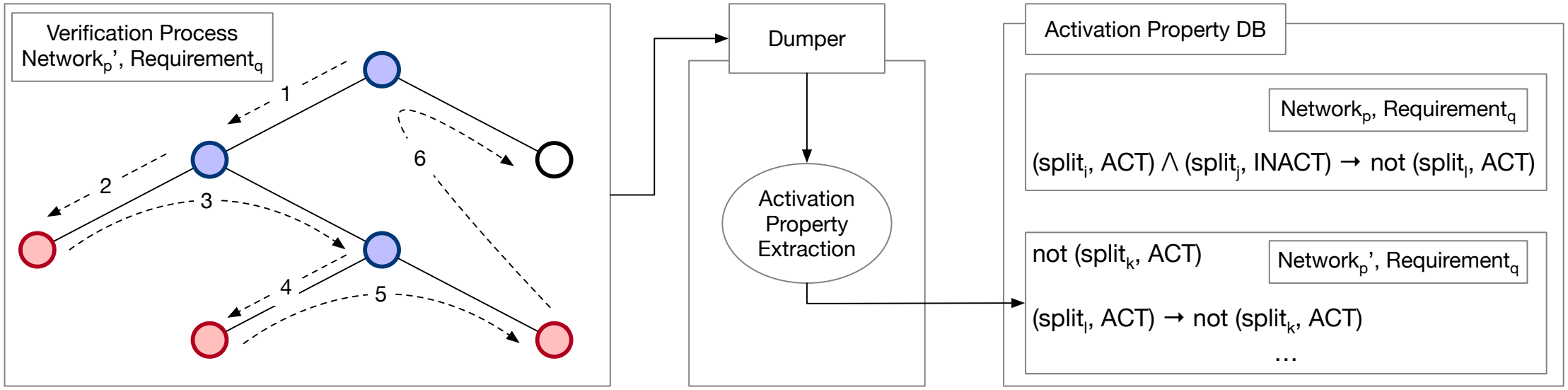
UNSAT Scenario: { ((layer$_i$, node$_j$), INACT),
((layer$_m$, node$_n$), INACT) }

Activation Property: ((layer$_i$, node$_j$), INACT) → not ((layer$_m$, node$_n$), INACT)

DB Generating Phase: Populating the DB

- 추출한 Activation Property를 프레임워크 DB에 저장할 수 있도록 지원하는 Dumper 인터페이스 제공

  → 현재 검증 중인 신경망과 요구사항별 activation property 저장

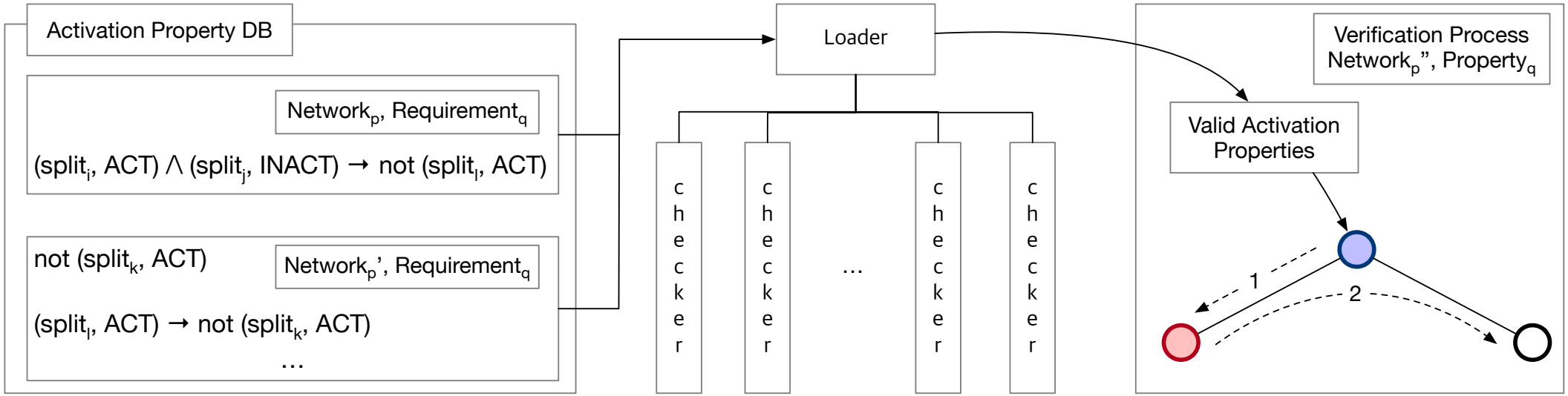


DB Utilization Phase: Parallel Validity Check & Loading DB

- 검증 시작 전, 유효한 Activation Property를 DB로부터 로드할 수 있도록 하는 Loader 인터페이스 제공

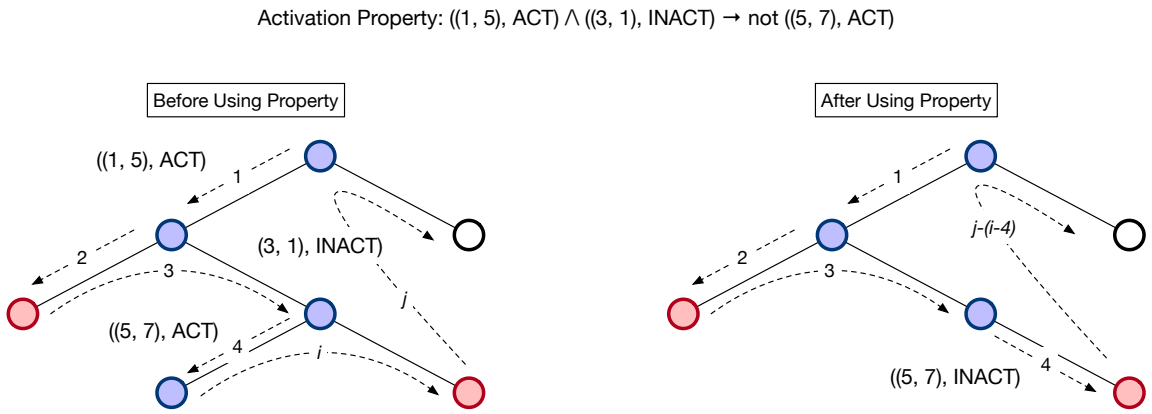
  → 변화된 신경망에 대해서도 activation property (:= UNSAT scenario)가 성립되는지 확인

  → 성능 최적화를 위해 병렬적으로 확인



DB Utilization Phase: Activation Property Usage

- Activation property를 위반하는 split을 진행하지 않음



- Activation property 로 인해 활성화 함수의 상태가 추가로 결정되는 경우, 함께 split

Activation Property 1: ((1, 5), ACT) ∧ ((3, 1), INACT) → not ((5, 7), ACT)

Activation Property 2: not ((5, 7), ACT) → ((4, 2), INACT)

((1, 5), ACT) 상태에서 ((3, 1), INACT) split이 진행된 경우, ((4, 2), INACT) split도 함께 진행

Experimental Setup

- 사용된 벤치마크

  → 심층 신경망: MNISTFC (L2, L4)

  → 250개의 이미지에 대해 local robustness ($\epsilon = 0.02$) 요구사항 검증

- 사용된 신경망 최적화 방식

  → Quantization (int8, int16)

  → Pruning (7%, 12%)

- 성능 측정 방식: DB가 구축된 문제들에 대해

  두 가지 수치 계산

$$time\ speedup = \frac{\sum verification\ time\ (not\ using\ technique)}{\sum verfication\ time\ (when\ using\ technique)}$$

$$tree\ size\ decrease = \frac{\sum tree\ size\ (not\ using\ technique)}{\sum tree\ size\ (when\ using\ technique)}$$

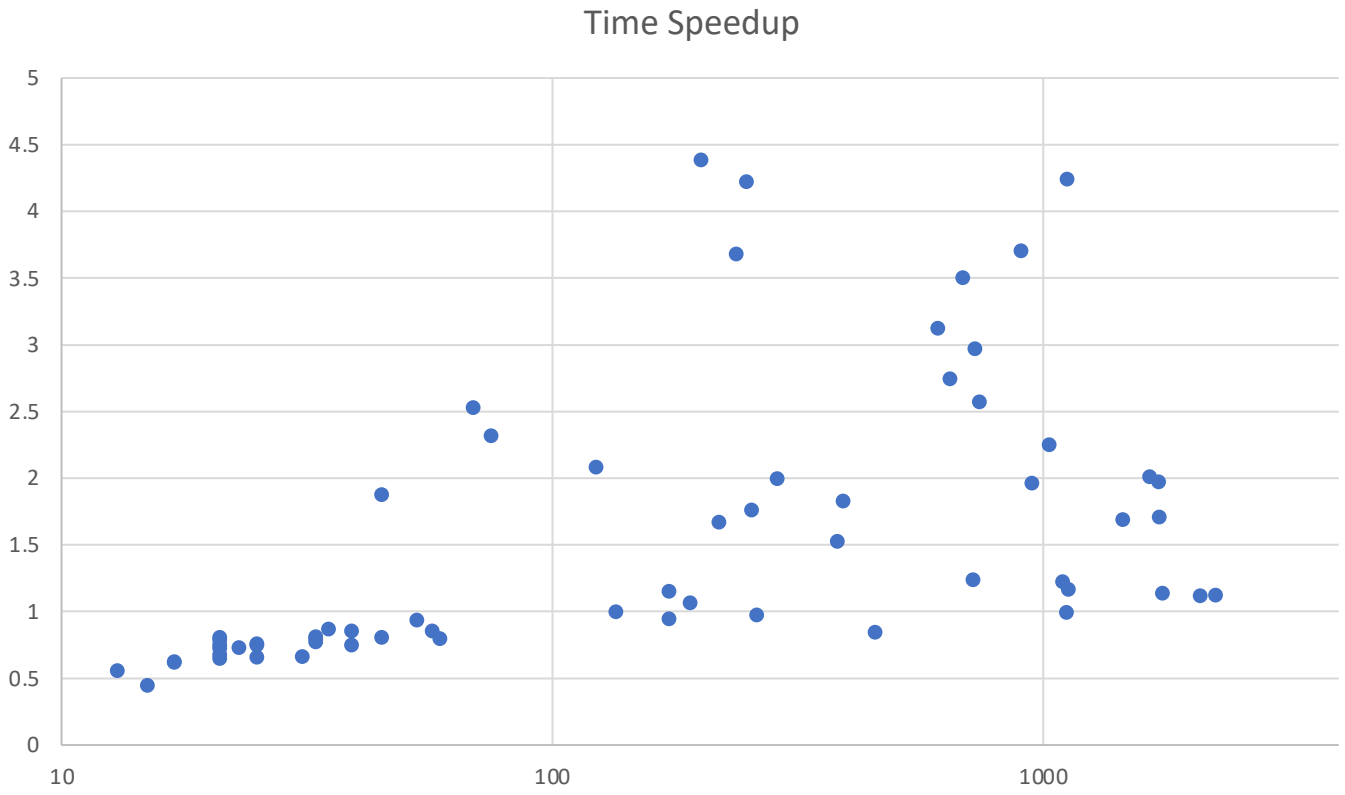
Effectiveness of Activation Property & Our Framework

- 모든 신경망 (L2, L4)과 최적화 조합에 대해 성능이 향상됨

	Optimization	**Tree Decrease**	**Time Speedup**
MNIST L2	Quantization.Int16	2.37	1.87
	Quantization.Int8	1.56	1.23
	Pruning 7%	2.24	2.05
	Pruning 12%	1.85	1.78
MNIST L4	Quantization.Int16	2.29	1.84
	Quantization.Int8	1.81	1.22
	Pruning 7%	1.91	1.63
	Pruning 12%	1.64	1.41

Better Performance for Harder Problems

- 탐색해야 하는 tree의 크기가 클수록, 즉 문제가 어려울수록 프레임워크 성능이 좋음



x-axis: 검증 문제 난이도 (프레임워크 미사용 시log scale tree size)

Comparison vs Previous Work (IVAN [PLDI 2023])

- MNISTFC (L4) 벤치마크에 대해 성능 비교

Optimization	**Tree Decrease**		**Time Speedup**	
	IVAN	**DB**	**IVAN**	**DB**
Quantization.Int16	0.99	2.29	1.33	1.84
Quantization.Int8	0.57	1.81	0.71	1.22
Pruning 7%	0.83	1.91	1.2	1.63
Pruning 12%	0.79	1.64	0.99	1.41

Ongoing Work

- 다양한 incremental 검증 기법을 포괄하고 확장할 수 있는 완성된 프레임워크 제공

  → 검증 과정 시작 전에, Activation Property이 만족됨을 확인

  → DB에 저장된/될 Activation Property 간 관계를 파악해서 DB 간소화 혹은 성질 강화

- 프레임워크에 SOTA non-incremental 검증기 $\alpha - \beta$ CROWN 도구 integrate

Thank You

Utilization of Activation Property DB for Efficient Incremental Neural Network Verification

채승현, 연주은

Software Verification Lab., Pohang University of Science and Engineering, South Korea

Introduction

- Incremental Neural Network Verification
  - 심층 신경망은 학습 후 곧바로 사용되지 않으며, 성능 향상을 위한 최적화 과정을 거치게 됨.
  - ⇒ 최적화를 통해 달라진 신경망에 대한 재검증이 필요!

  - 이전 검증 과정의 정보를 재사용하여 검증의 효율성을 높이는 연구.
- Abstraction and Iterative Refinement of Activation Functions

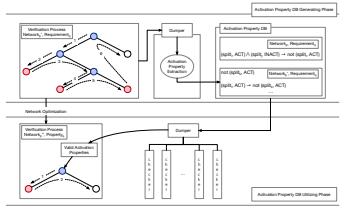
Motivation: Activation Property

- Activation Property

$$\bigwedge_i (split_i, ACTIVE) \wedge \bigwedge_j (split_j, INACTIVE) \rightarrow \neg(split_k, ACTIVE \text{ or } INACTIVE)$$

  - ACTIVE, INACTIVE split들이 특정 노드의 활성화 함수 상태를 논리적으로 imply하는 성질.
  - ⇒ 이후 검증 문제들에서도 해당 성질이 유지될 가능성이 높음.
- 연구 목표
  - 이전 검증 과정에서 추출한 activation property들을 DB로 구축하고,
  - 이후 검증 과정에서 로딩하여 활용함으로써 검증 성능 향상.

Activation Property DB utilizing framework

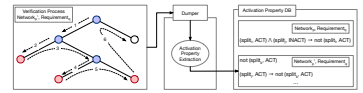


- 두 가지의 Phase로 구분 가능.
  1. Activation Property DB Generating Phase
  2. Activation Property DB Utilization Phase

Activation Property DB Generating Phase

- Trasnforming UNSAT Scenario to Activation Property

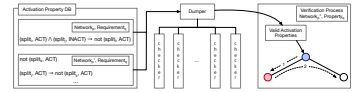
  - ⇒ UNSAT 결과를 유발한 split들의 집합, UNSAT scenario를 추출한 뒤, activation property로 변환.
- Populating the DB



  - ⇒ 신경망과 요구사항 마다 activation property를 저장.

Activation Property DB Utilization Phase

- Parallel Validity Check & Loading the DB



  - ⇒ 검증 시작 시 사용 가능한 유효한 activation property를 필터링하여 로딩한 후, 검증 과정에 활용.
- DB Utilization
  - Activation property를 위반하는 split을 진행하지 않음.
  - Activation property로 활성화 함수의 상태가 결정되는 노드가 있을 경우, 해당 노드를 기존 split과 함께 split.

Experimental Results

- 다양한 최적화 및 신경망에 대한 성능

	Optimization	Tree Decrease	Time Speedup
MNIST L2	Quantization.Int16	1.6	1.48
	Quantization.Int8	1.28	1.16
	Pruning 7%	1.34	1.56
MNIST L4	Quantization.Int16	1.03	1.14
	Quantization.Int8	1.13	1.07
	Pruning 7%	1.28	1.12

- 기존 incremental 검증 기법 IVAN과의 비교 (MNIST L4)

Optimization	Tree Decrease		Time Speedup	
	IVAN	DB	IVAN	DB
Quantization.Int16	0.8	1.03	1.01	1.14
Quantization.Int8	0.84	1.13	0.84	1.07
Pruning 7%	0.87	1.28	1	1.12

Ongoing Work

- Activation property 활용 방식 추가 고안
- IVAN 외 다른 incremental verification 도구와의 비교 진행
- 프레임워크에 α-β CROWN도 추가

Contact: shchae7@postech.ac.kr