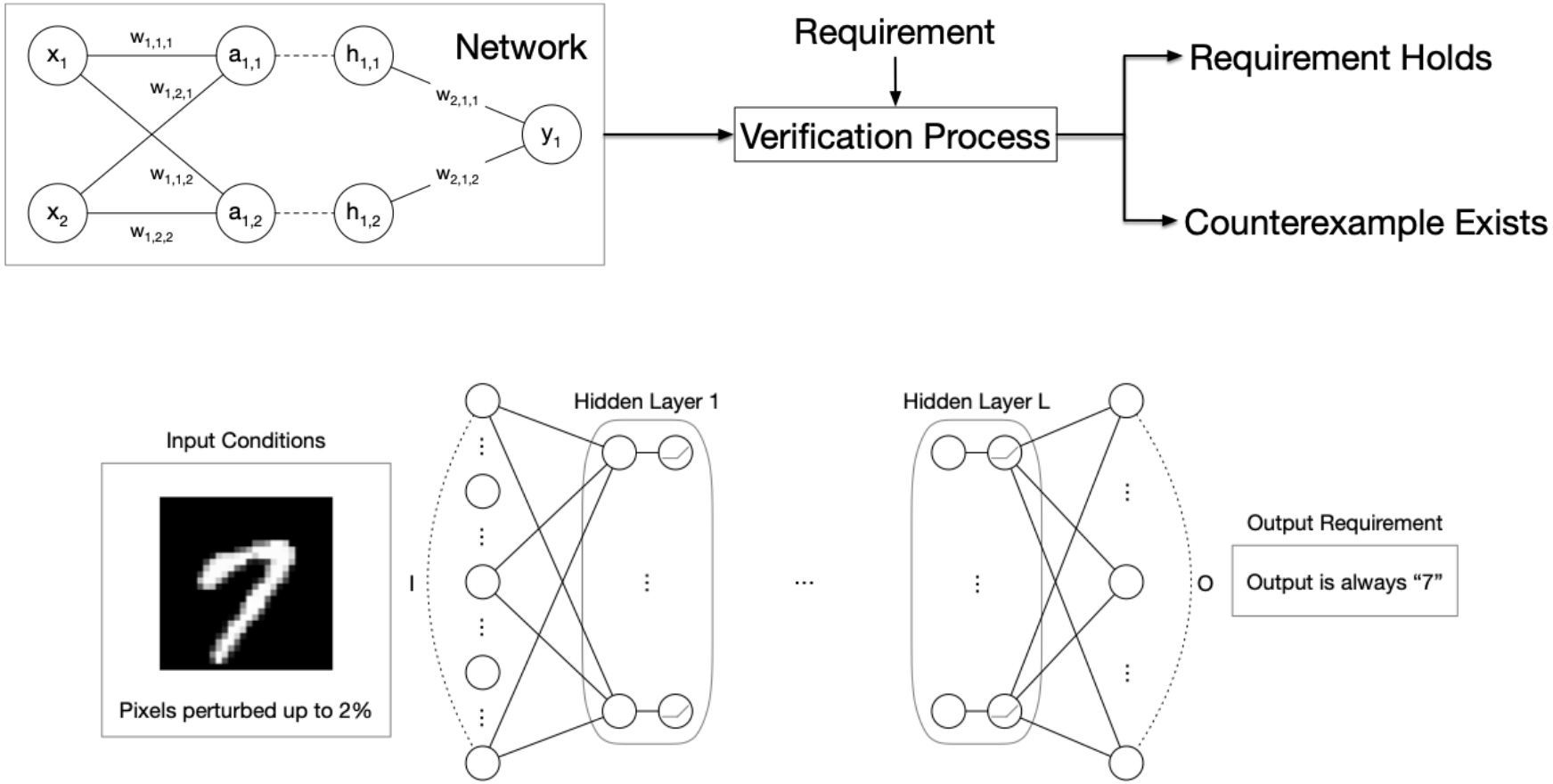
Incremental NN Verification with Implementation-Agnostic Activation Property DB

채승현

POSTECH Software Verification Lab

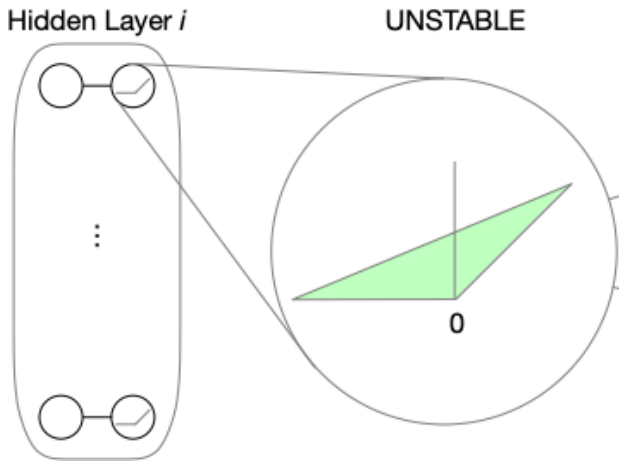
2026년 STAAR 겨울 정기 워크샵

Verification of Neural Networks



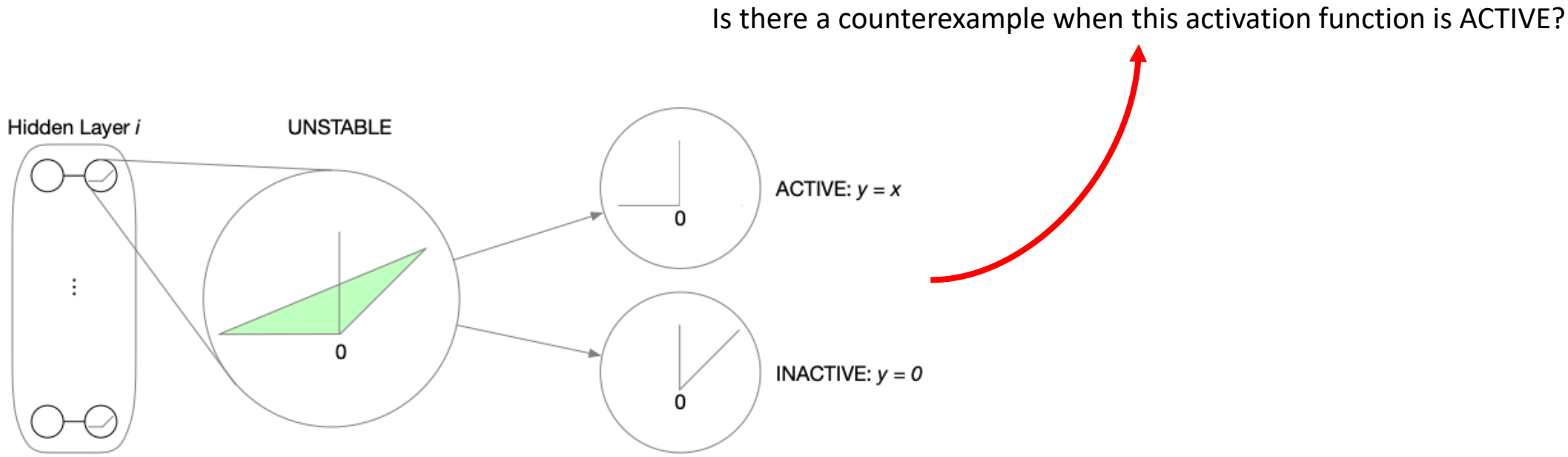
Activation Func. Refinement-based Verification

- 신경망을 구성하는 모든 활성화 함수를 선형 구조로 추상화



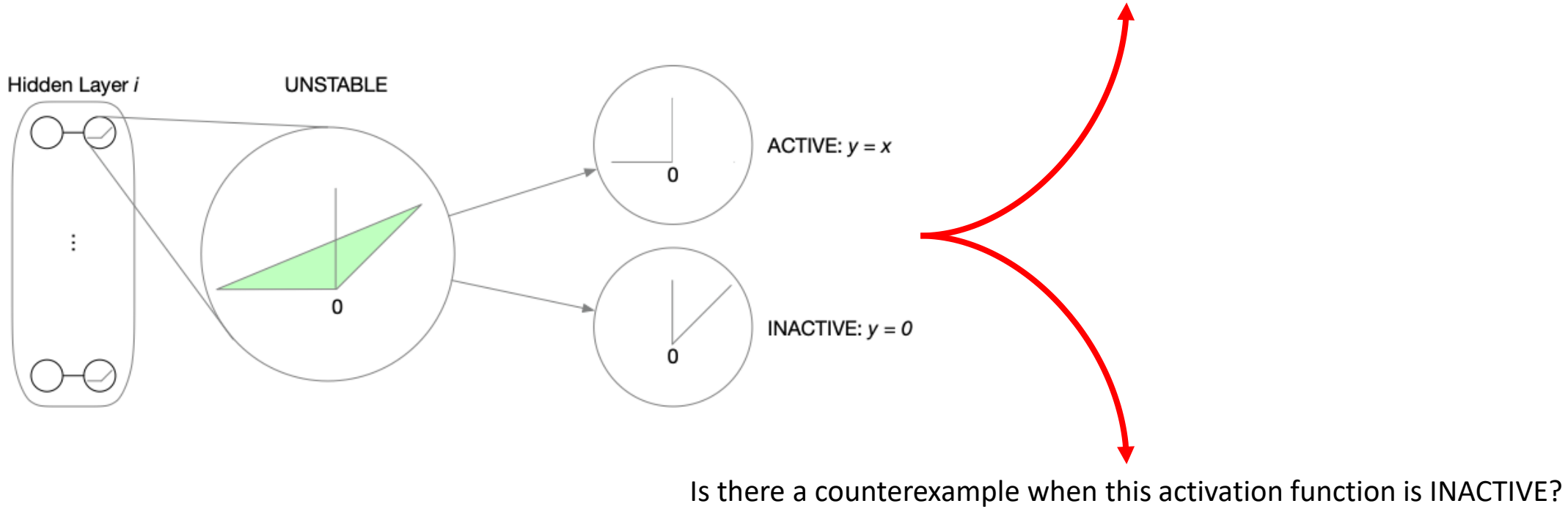
Activation Func. Refinement-based Verification

- 추상화된 활성화 함수 하나씩 선택하여, 선형적인 행동을 보이도록 상태를 고정: “refine/split”



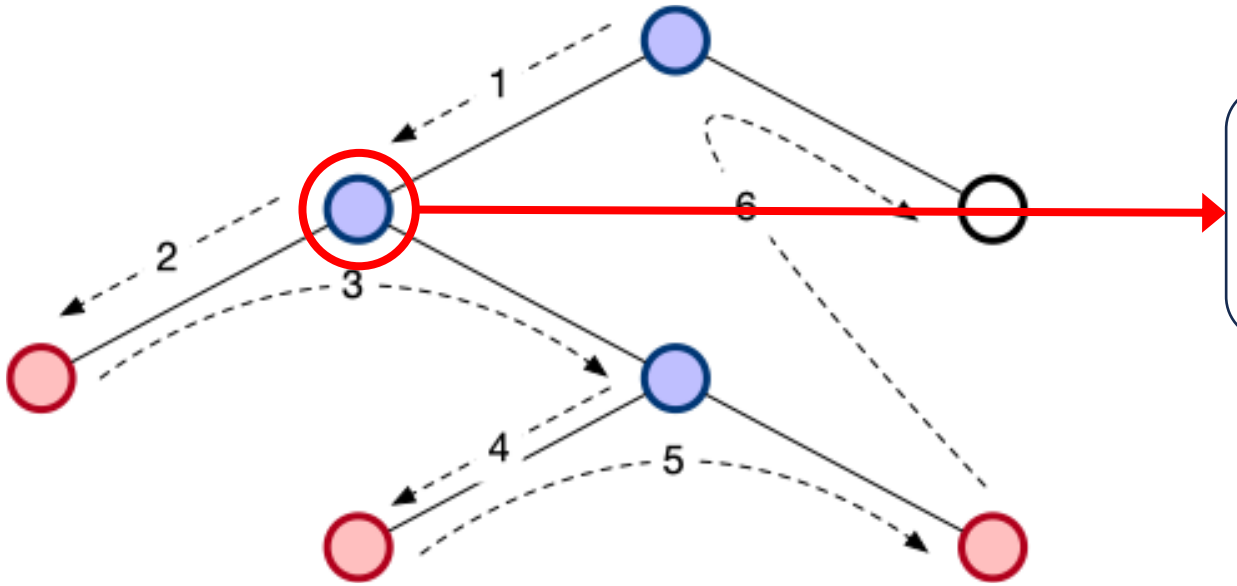
Activation Func. Refinement-based Verification

- 추상화된 활성화 함수 하나씩 선택하여, 선형적인 행동을 보이도록 상태를 고정: “refine/split”



Activation Func. Refinement-based Verification

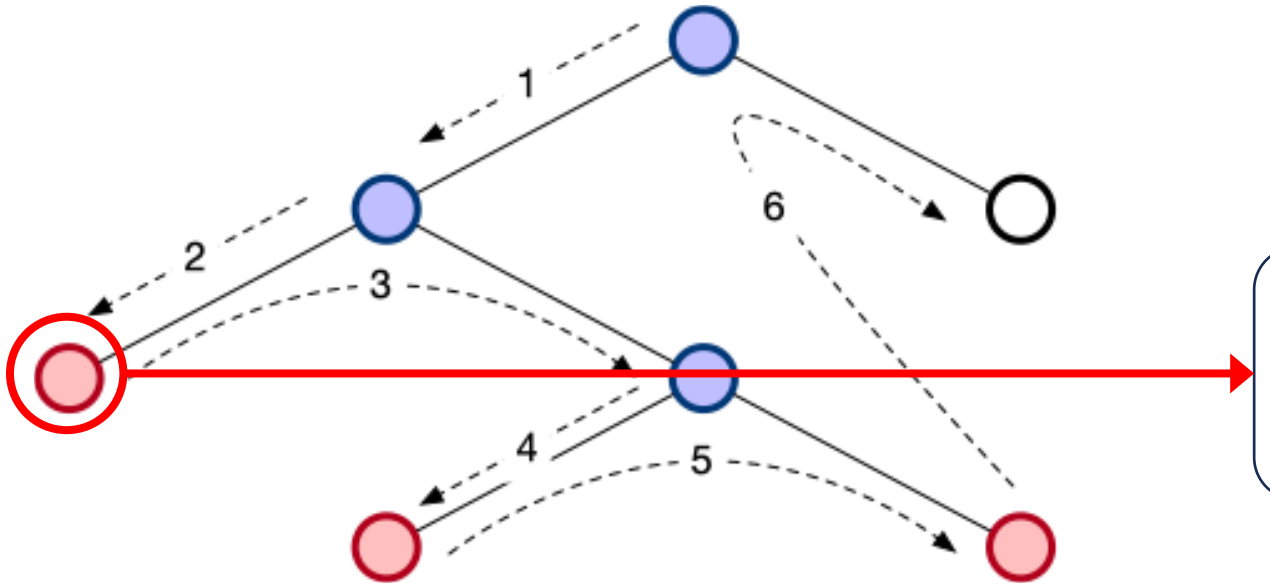
- 해당 과정에서 발생하는 (활성화 함수들의 상태로 정의 가능한) 문제 공간 탐색



Is there a counterexample when
- act. func. at (l_1, n_1) is *ACTIVE* ?

Activation Func. Refinement-based Verification

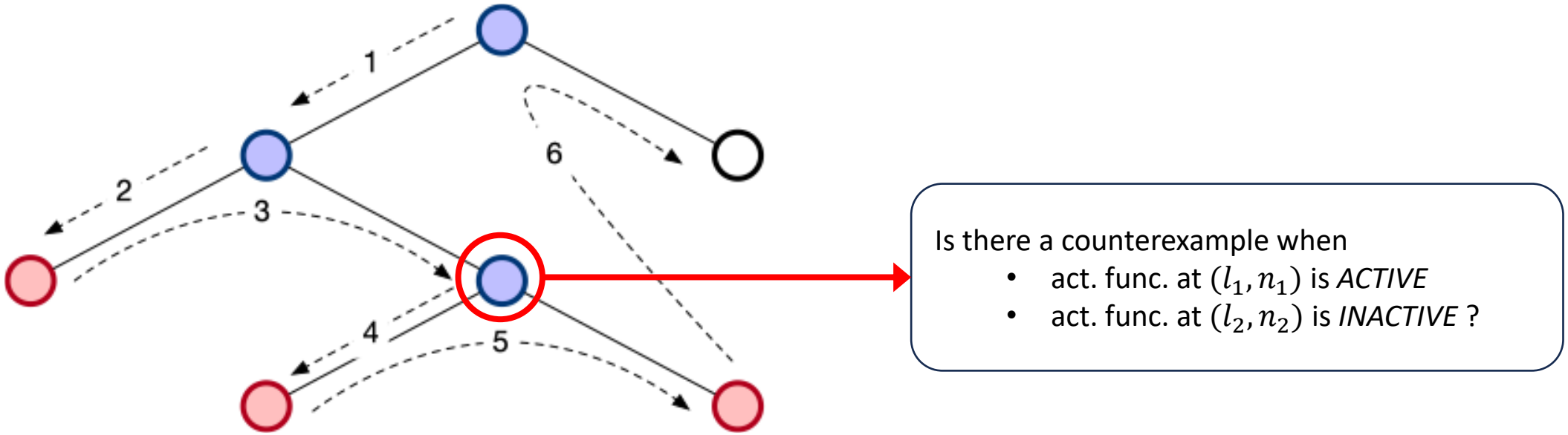
- 해당 과정에서 발생하는 (활성화 함수들의 상태로 정의 가능한) 문제 공간 탐색



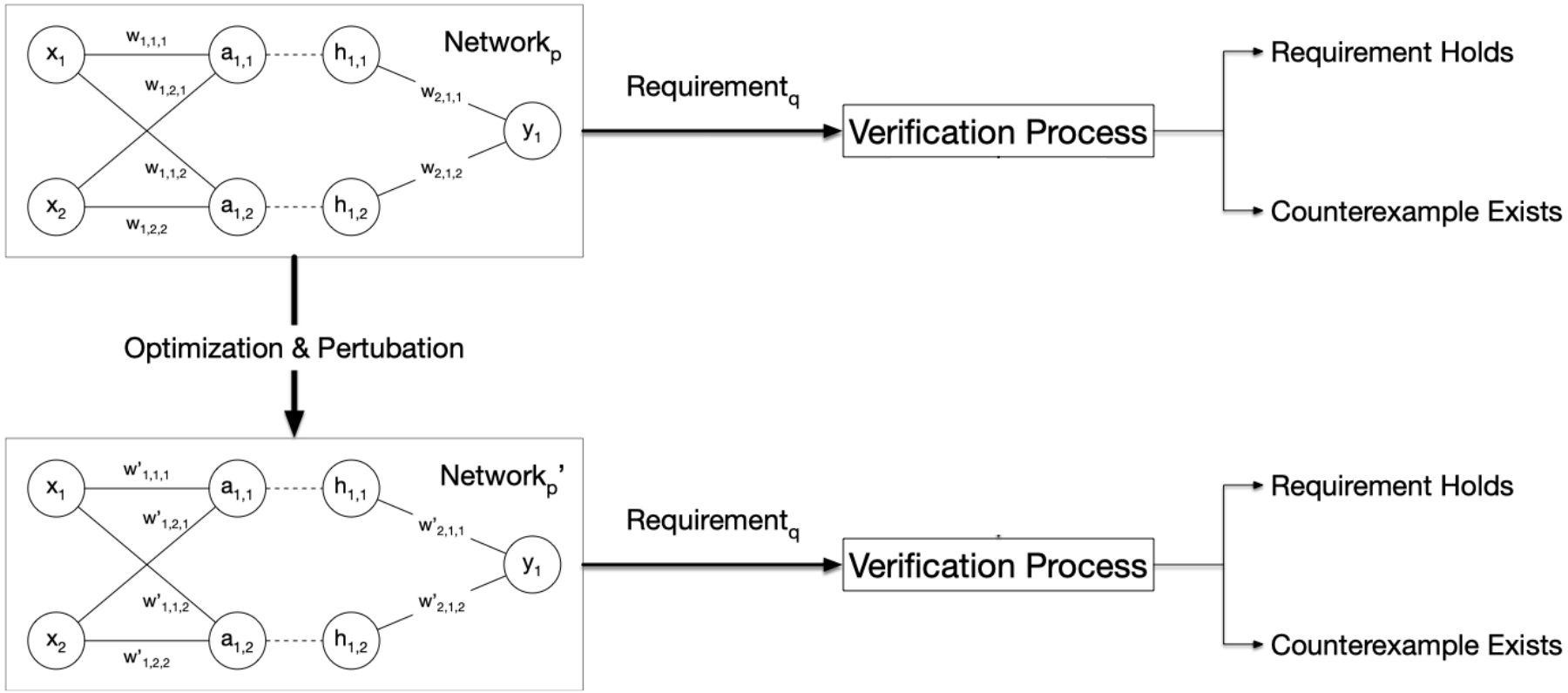
Is there a counterexample when
- act. func. at (l_1, n_1) is *ACTIVE*
- act. func. at (l_2, n_2) is *ACTIVE* ?

Activation Func. Refinement-based Verification

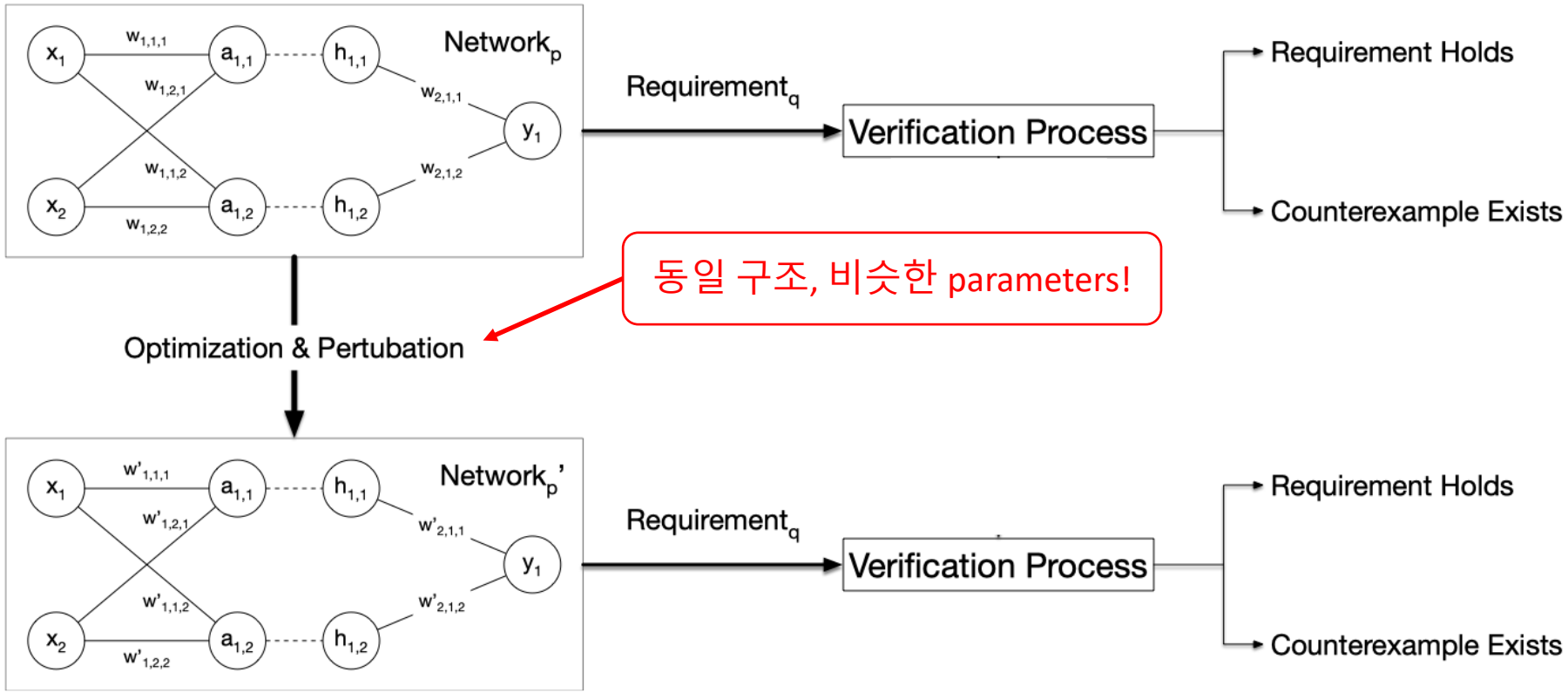
- 해당 과정에서 발생하는 (활성화 함수들의 상태로 정의 가능한) 문제 공간 탐색



Incremental Verification of Neural Networks

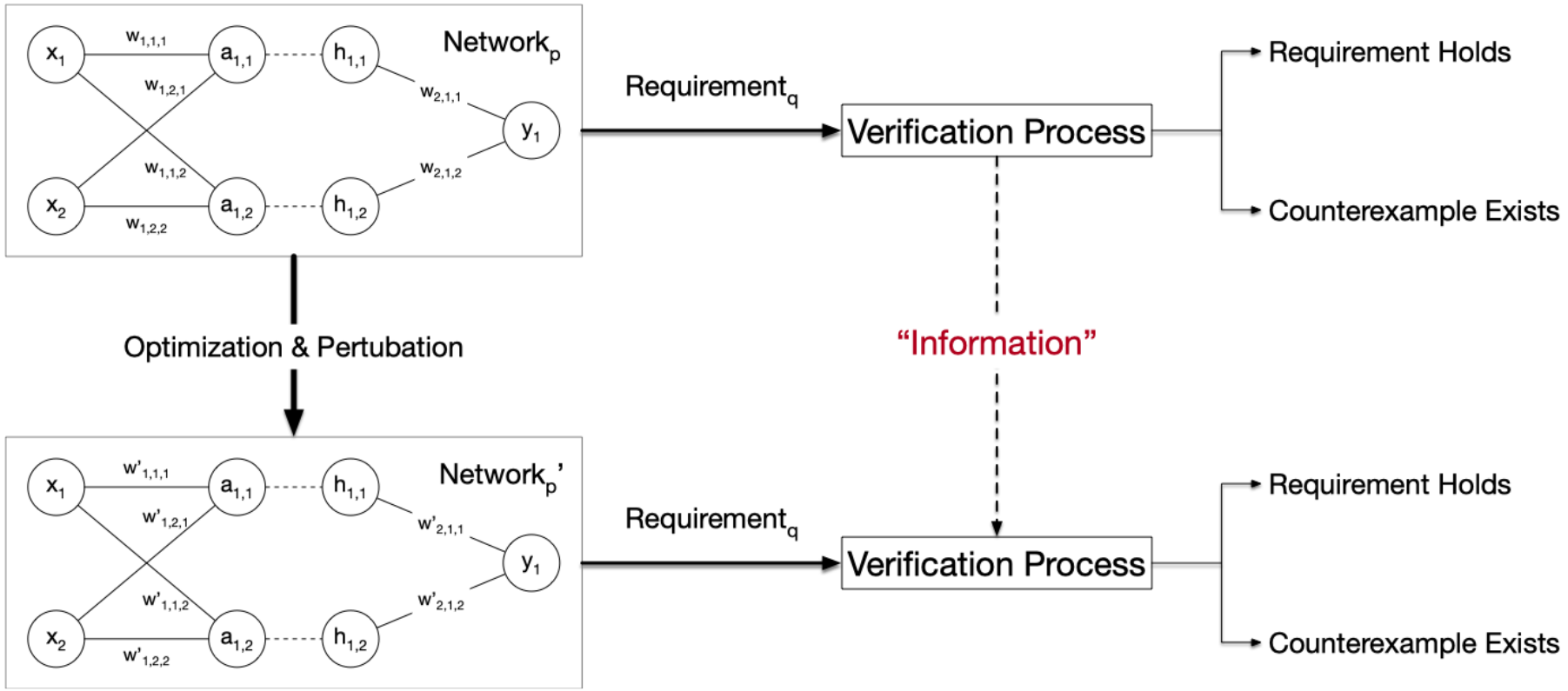


Incremental Verification of Neural Networks



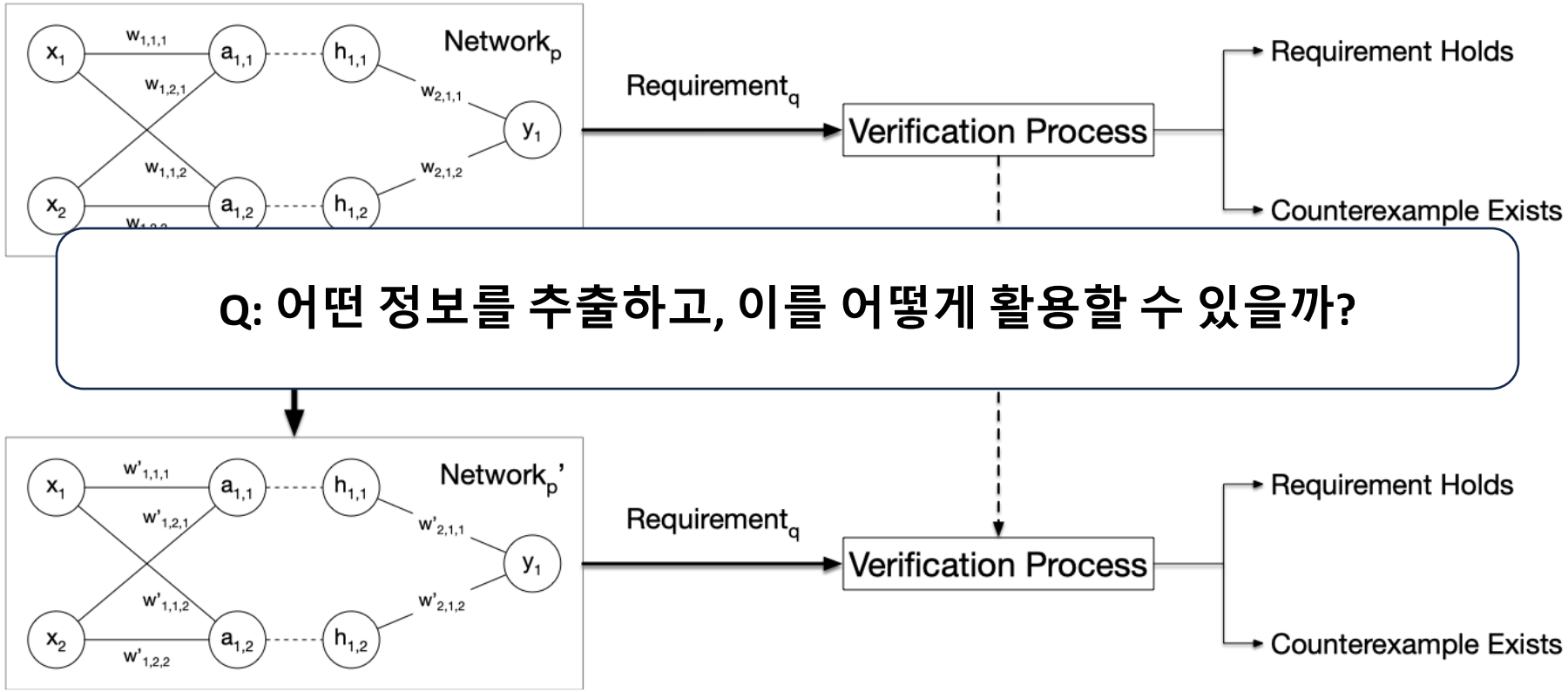
Incremental Verification of Neural Networks

Q: 이전 검증 과정의 정보를 활용할 수 있을까?



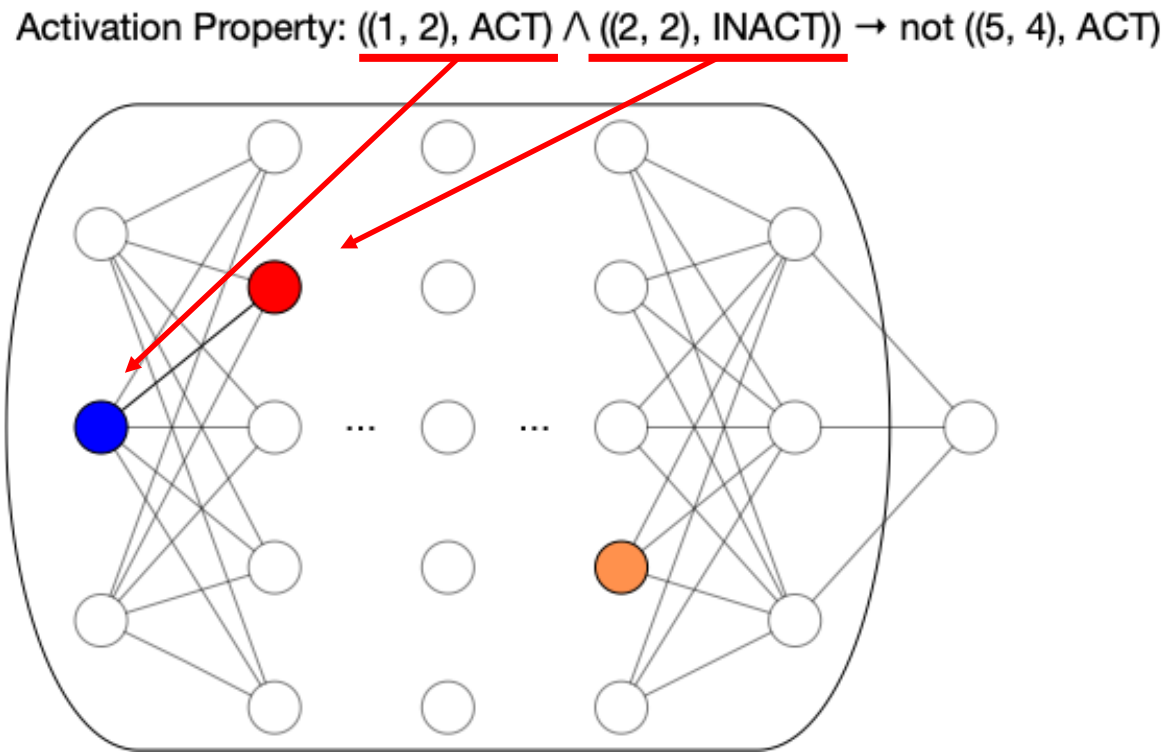
Incremental Verification of Neural Networks

Q: 이전 검증 과정의 정보를 활용할 수 있을까?



A: Proposing “Activation Property” (Act. Prop.)

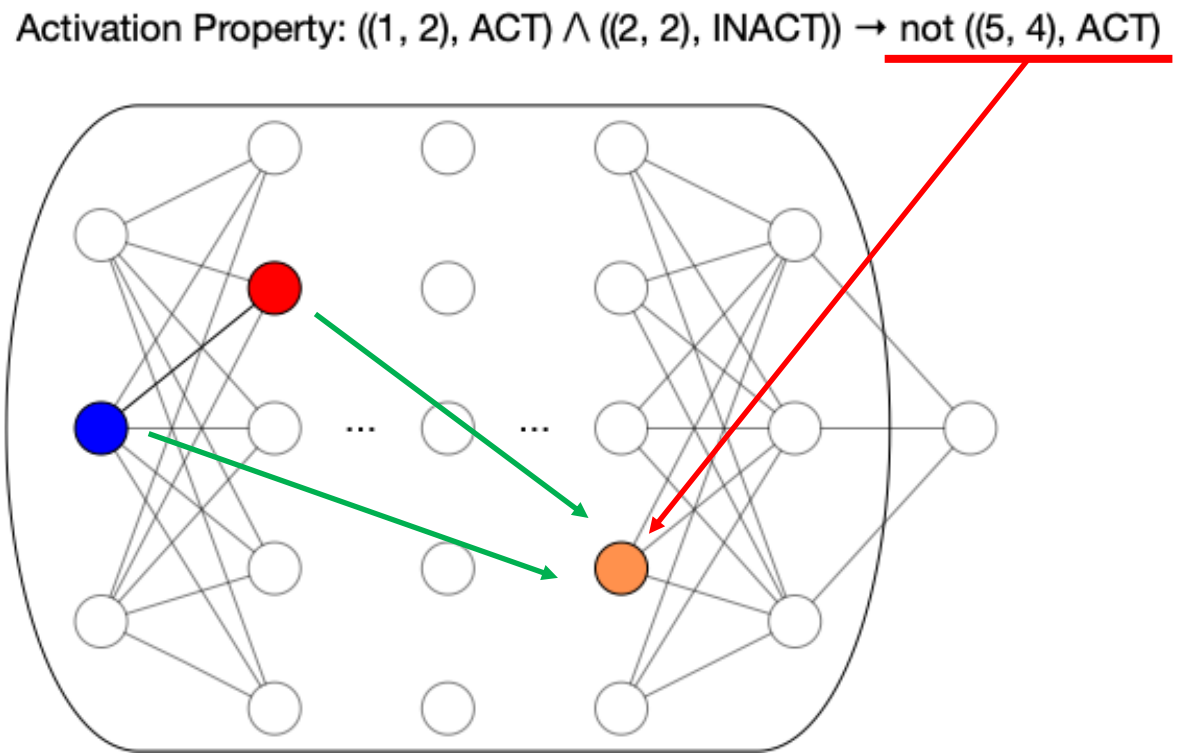
- Activation property (활성화 성질): 특정 활성화 함수들의 상태 조합이 다른 활성화 함수의 상태를 제한하는 성질



(1, 2), (2,2) 위치의 활성화 함수가
각각 ACTIVE, INACTIVE이면,

A: Proposing “Activation Property” (Act. Prop.)

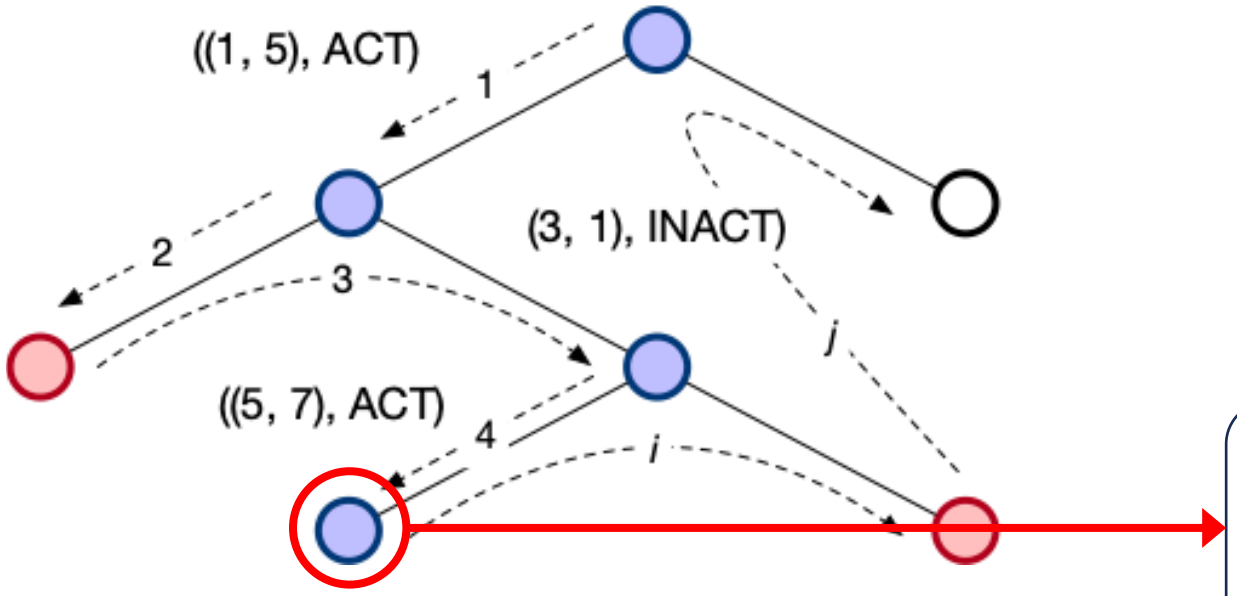
- Activation property (활성화 성질): 특정 활성화 함수들의 상태 조합이 다른 활성화 함수의 상태를 제한하는 성질



(5,4) 위치의 활성화 함수는 ACTIVE일 수 없음

Utility of Activation Property

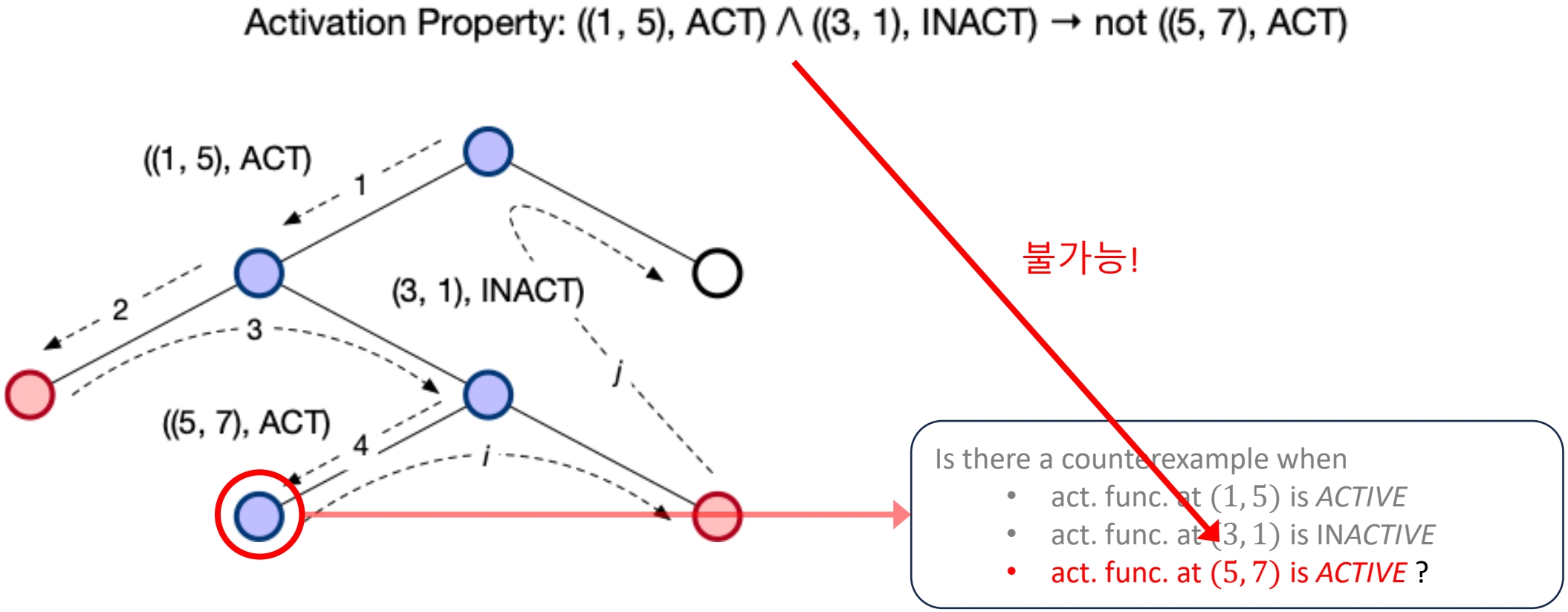
Activation Property: ((1, 5), ACT) ∧ ((3, 1), INACT) → not ((5, 7), ACT)



Is there a counterexample when

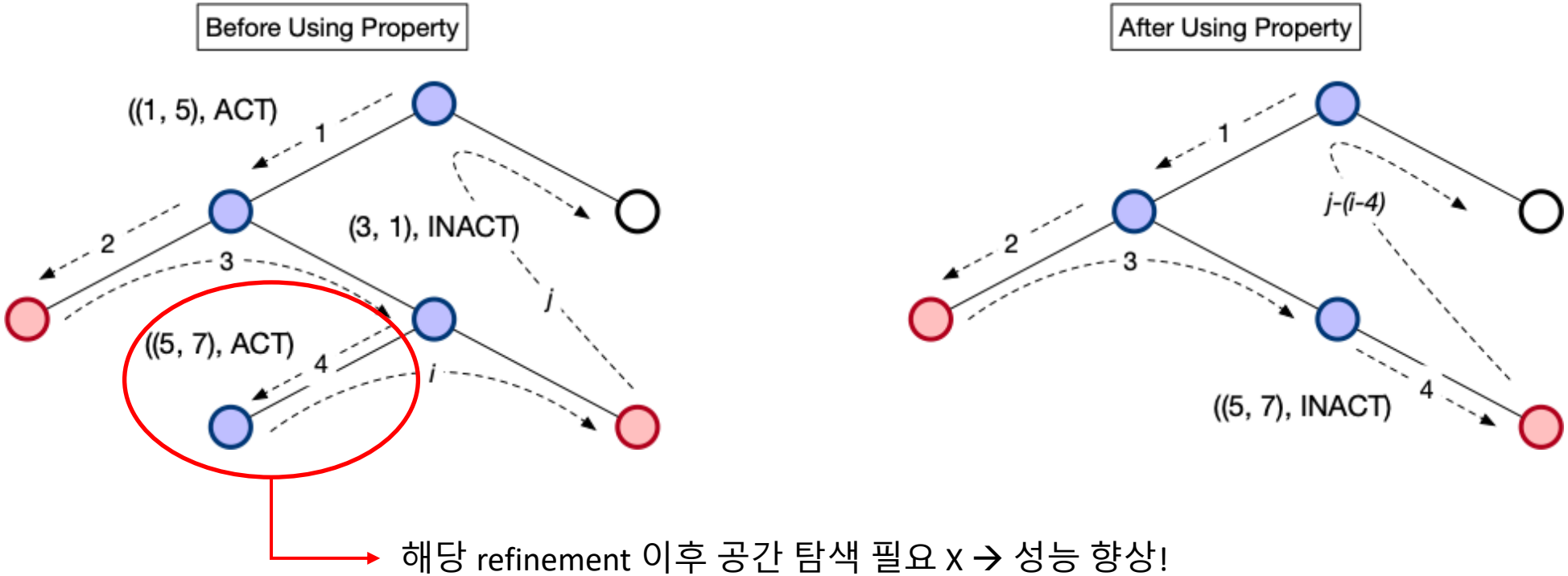
- act. func. at $(1, 5)$ is *ACTIVE*
- act. func. at $(3, 1)$ is IN*ACTIVE*
- act. func. at $(5, 7)$ is *ACTIVE* ?

Utility of Activation Property



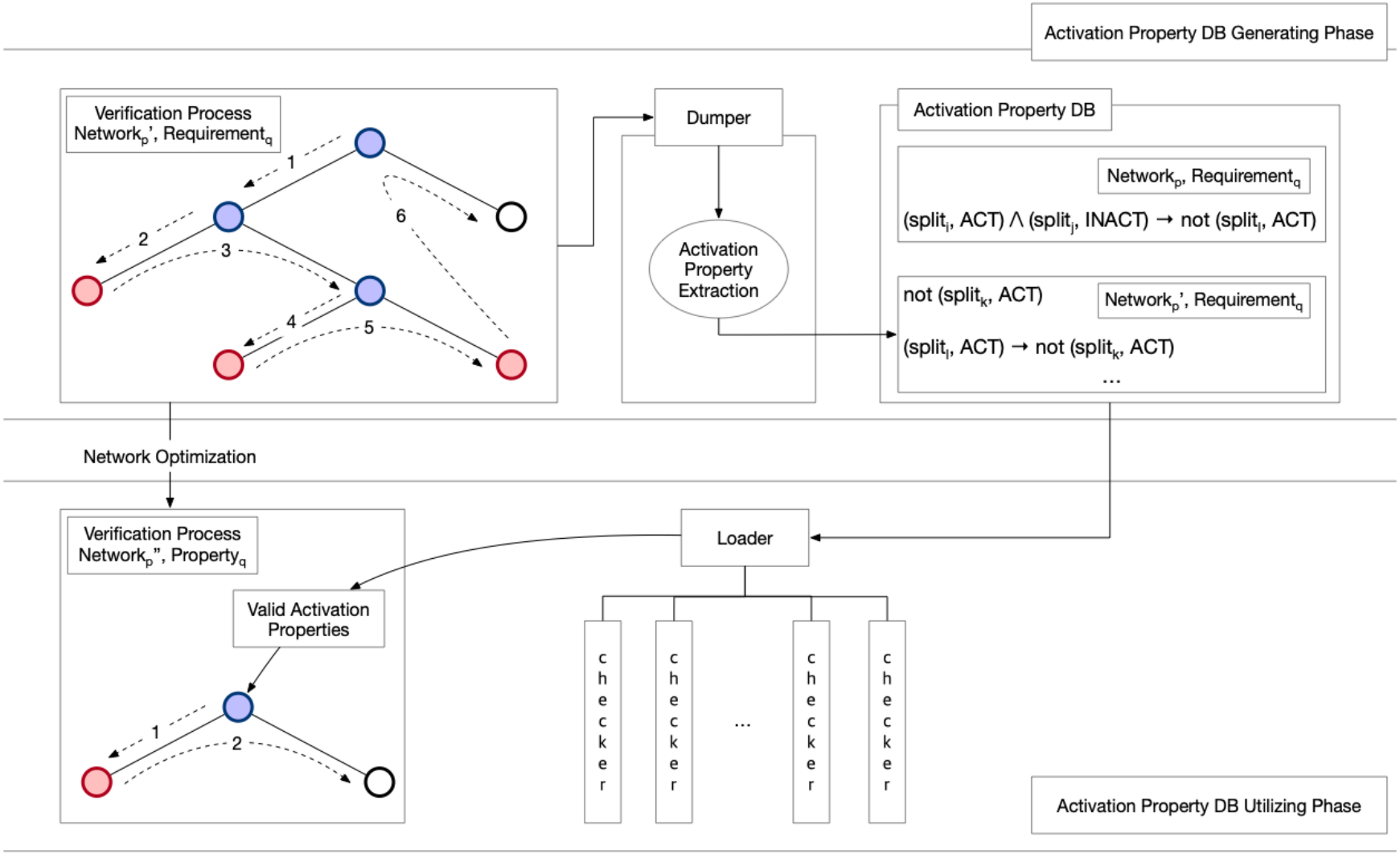
Utility of Activation Property

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

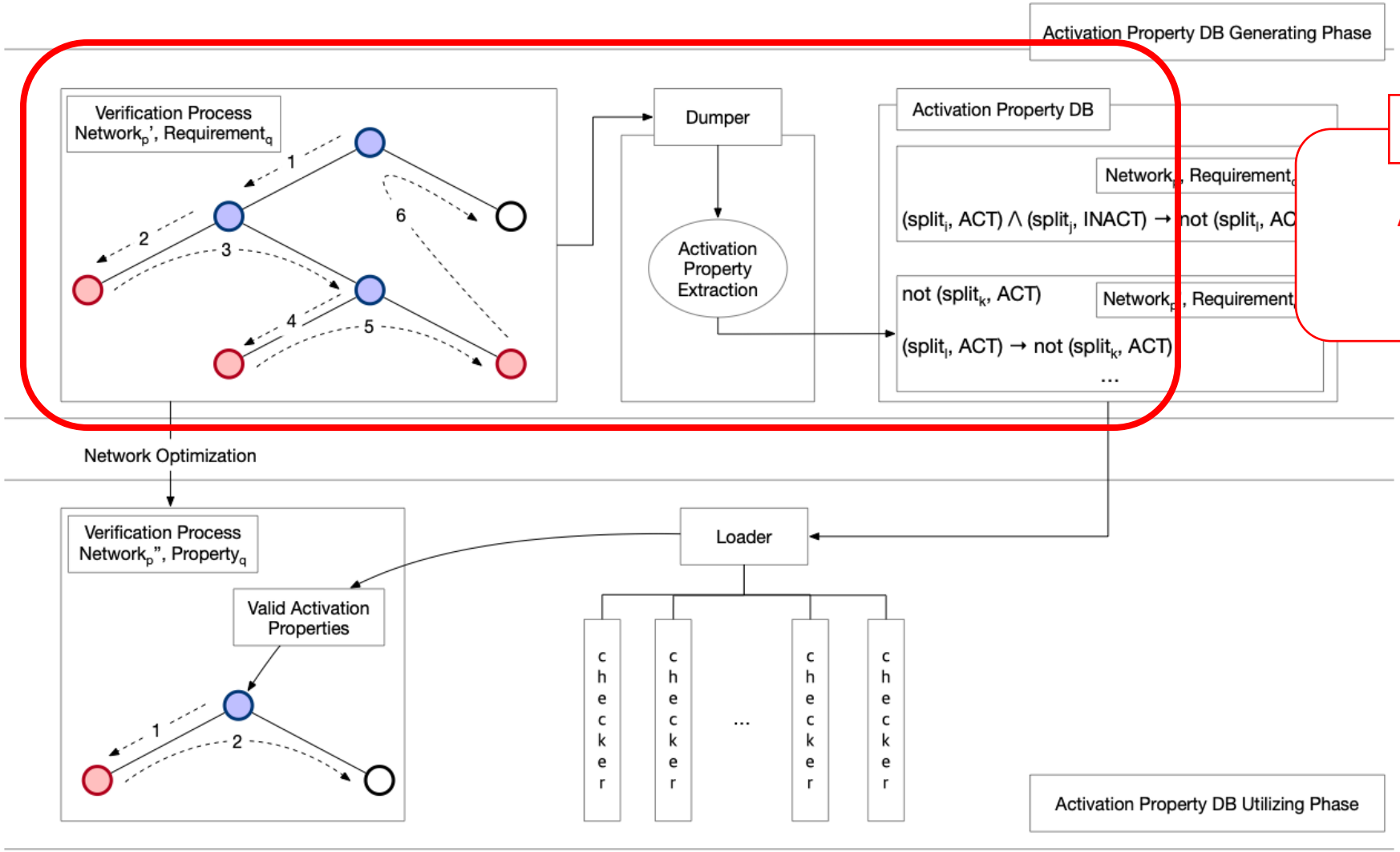


해당 refinement 이후 공간 탐색 필요 X → 성능 향상!

Activation Property DB Utilizing Framework Overview



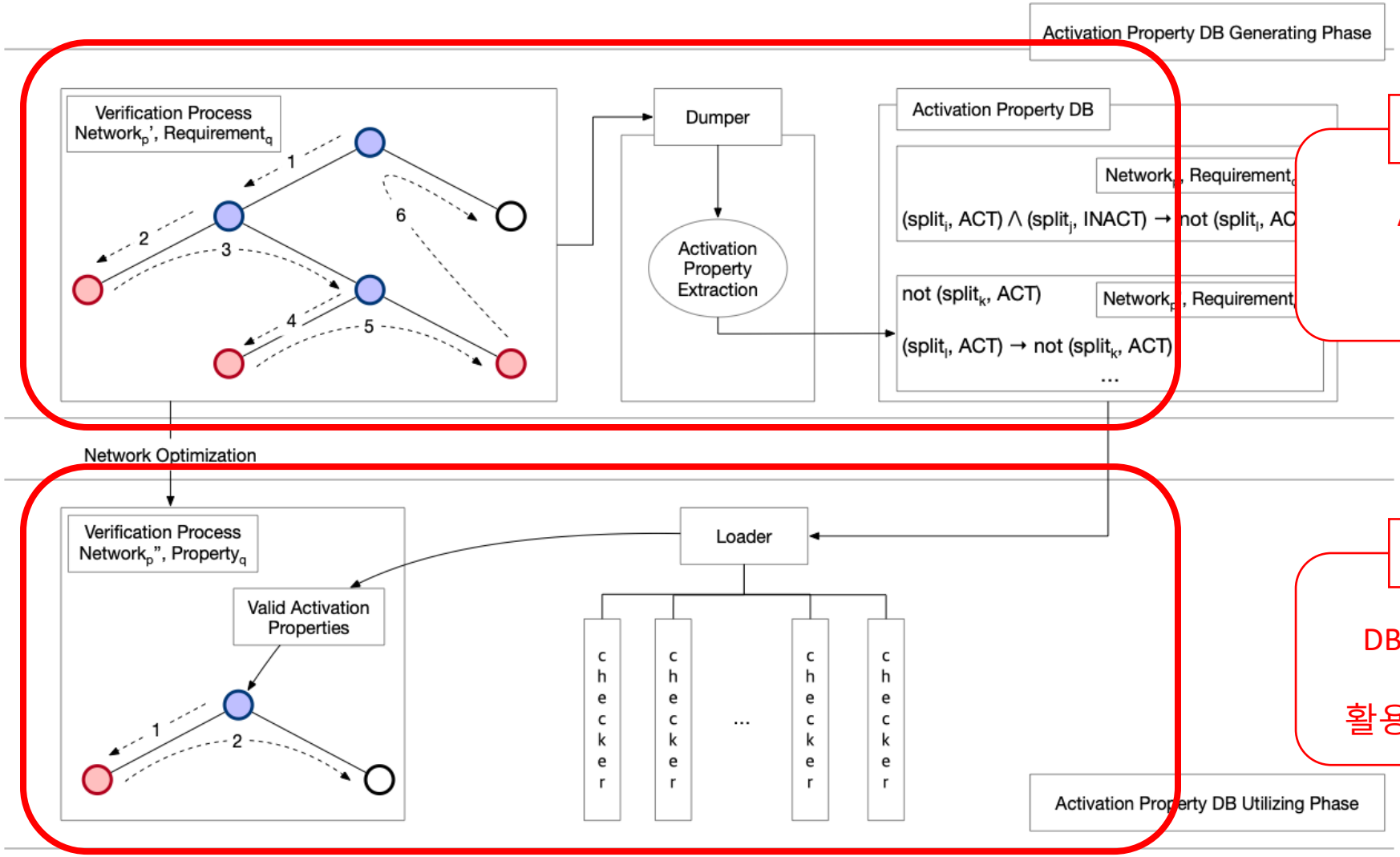
Activation Property DB Utilizing Framework Overview



이전 검증 과정

Act. Prop. 추출/계산
+
DB에 저장

Activation Property DB Utilizing Framework Overview



이전 검증 과정

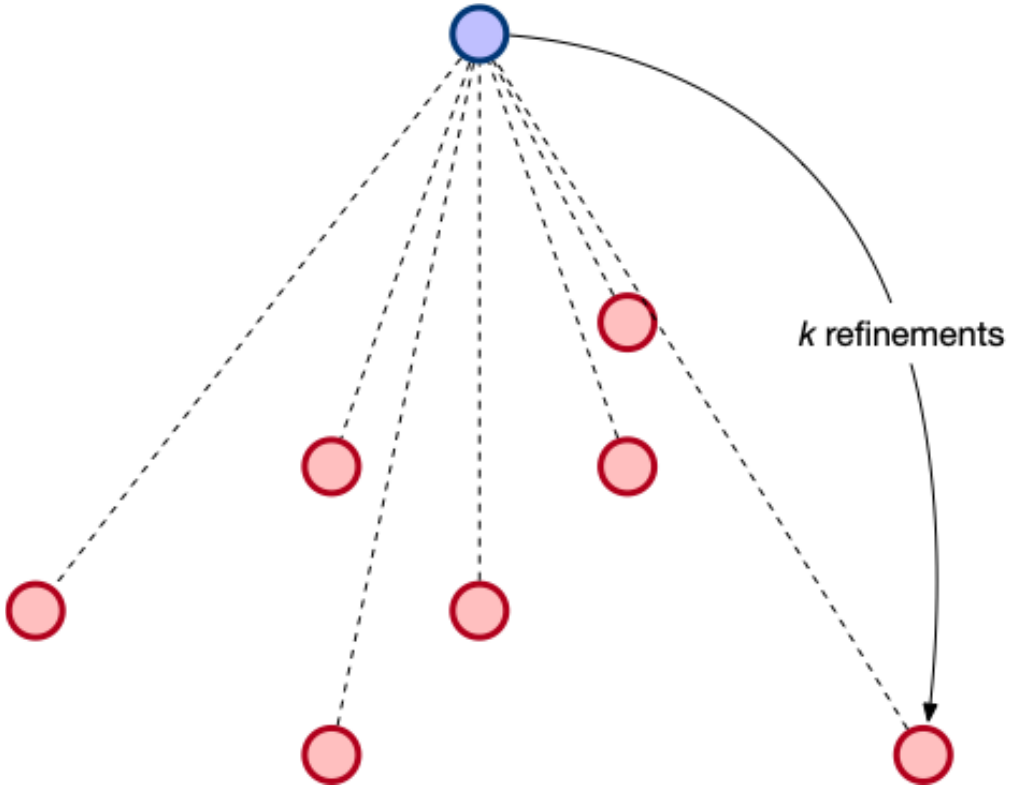
Act. Prop. 추출/계산
+
DB에 저장

이후 검증 과정

DB로부터 Act. Prop. 로딩
+
활용하여 검증 과정 가속화

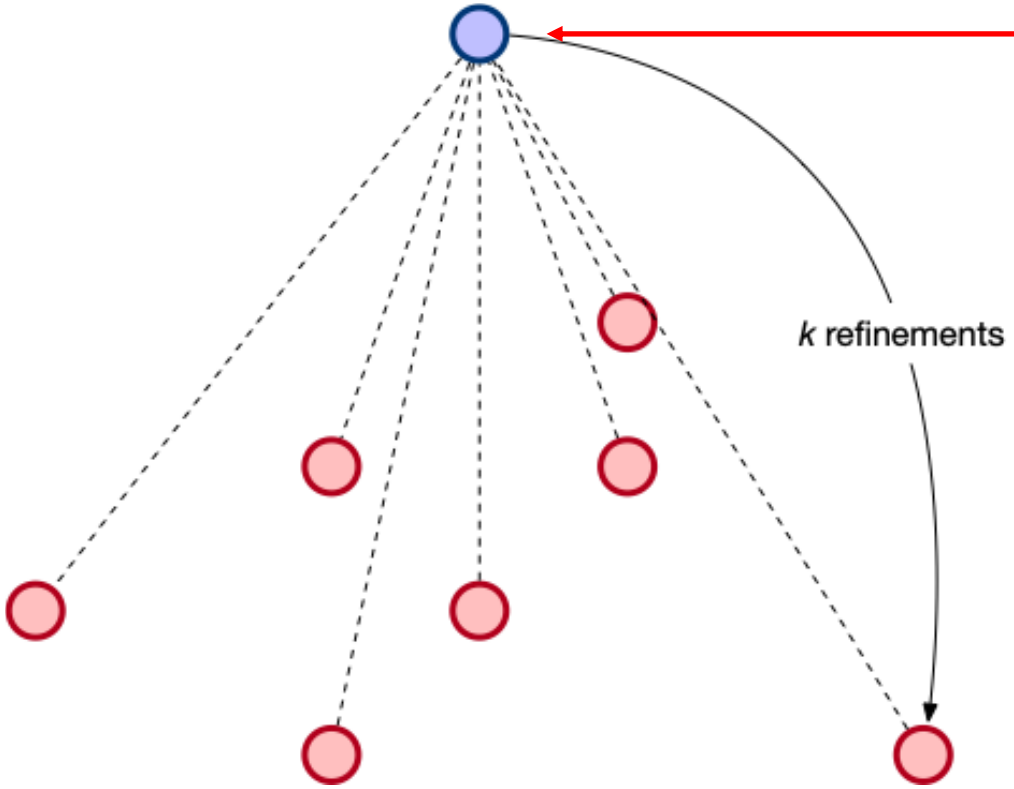
Factors Degrading Act. Prop. Usability

- Act. Prop.은

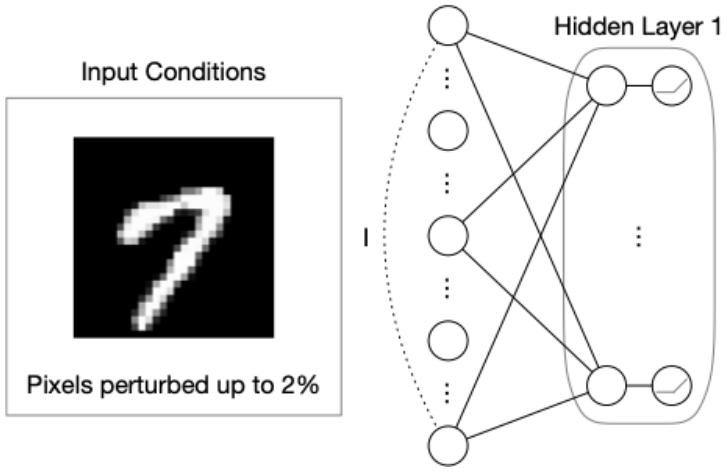


Factors Degrading Act. Prop. Usability

- Act. Prop.은

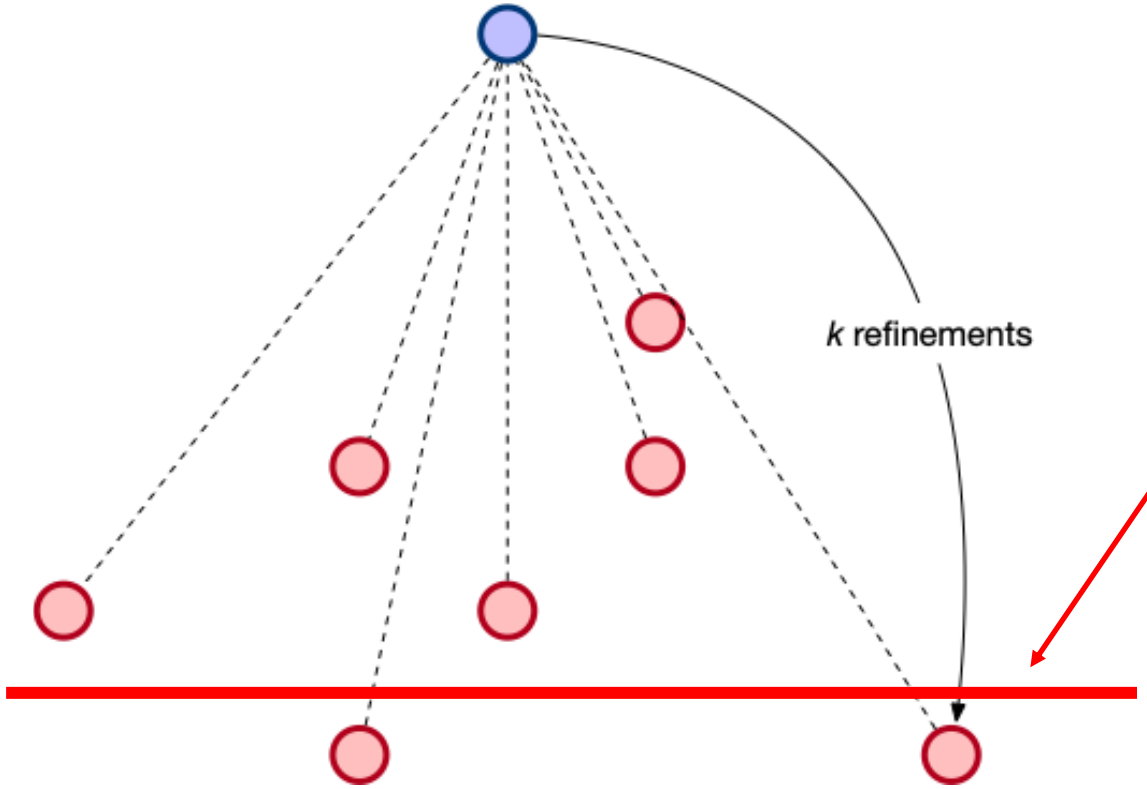


검증하고자 했던 요구사항의
입력 조건 하에서만 유효



Factors Degrading Act. Prop. Usability

- Act. Prop.은

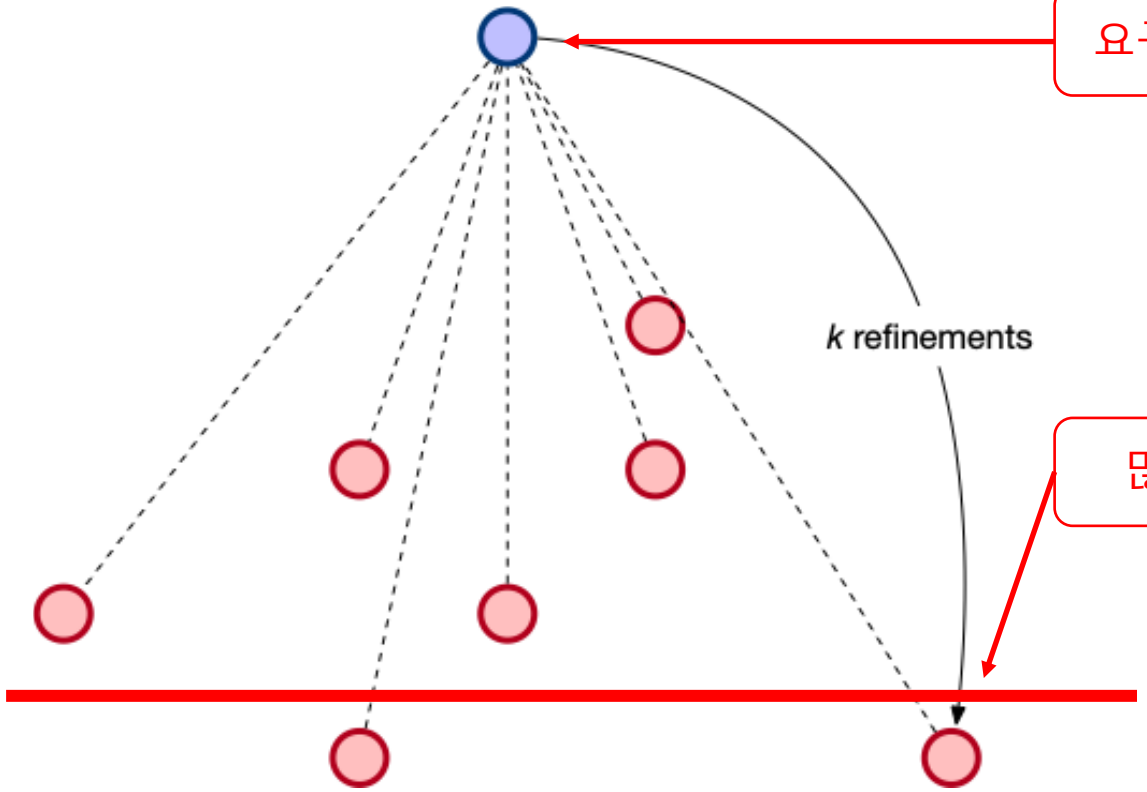


보통 많은 refinement가
이뤄진 후에만 발견

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

Enhancing Act. Prop. Usability

- Act. Prop.은



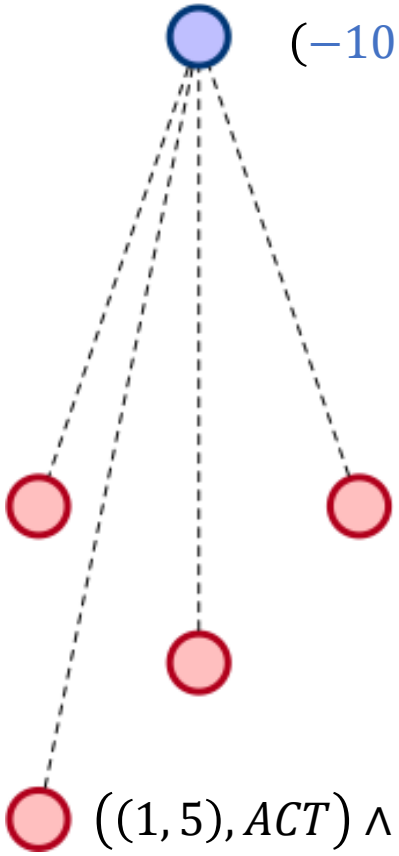
요구사항의 입력 조건에 귀속

➔ IO Generalization

많은 refinement 가정 필요

➔ Split Scenario Generalization

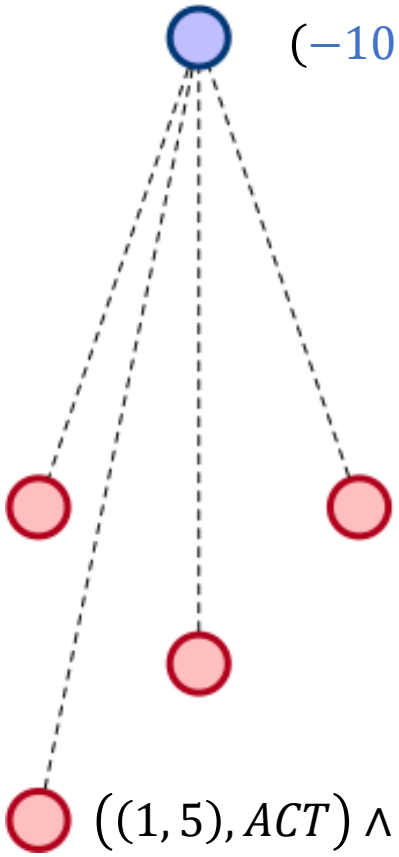
Enhancing Act. Prop. Usability: IO Generalization



$(-10 \leq x_1 \leq 10) \wedge (2 \leq x_2 \leq 5) \wedge \cdots$

$\big((1,5), ACT\big) \wedge \big((3,1), INACT\big) \wedge \big((4,1), ACT\big) \rightarrow \neg((5,7), ACT)$

Enhancing Act. Prop. Usability: IO Generalization



$(-10 \leq x_1 \leq 10) \wedge (2 \leq x_2 \leq 5) \wedge \cdots \longrightarrow (-\infty \leq x_1 \leq \infty) \wedge (2 \leq x_2 \leq 5) \wedge \cdots$

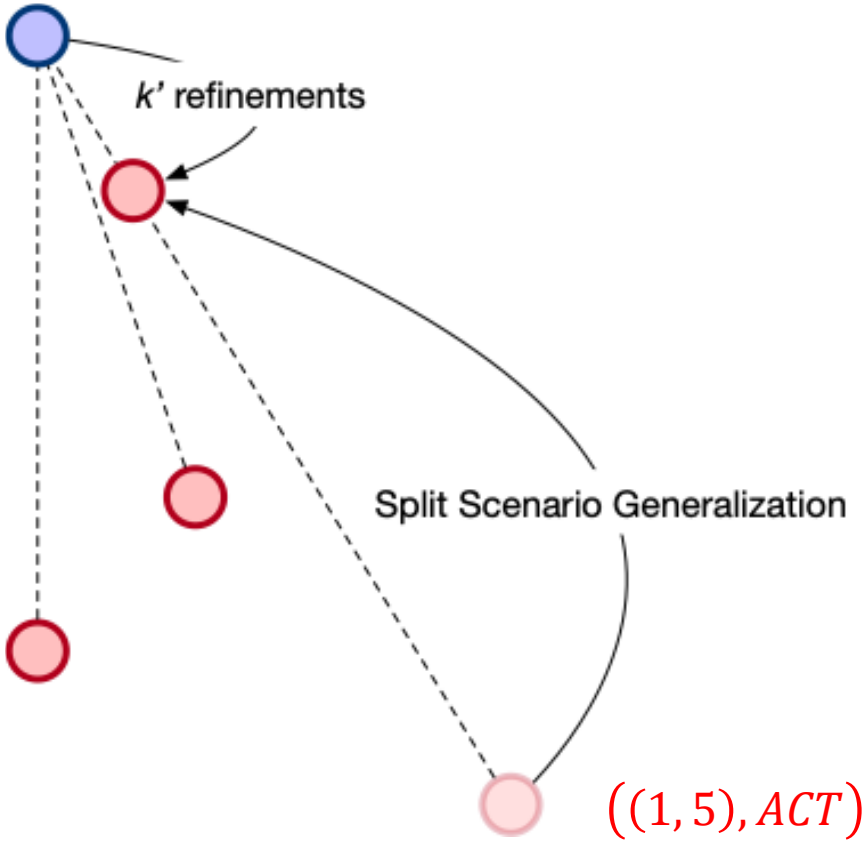
유효한 입력 조건 확장

→

다른 입력 조건을 가진 요구사항 검증 과정에서도 사용 가능!

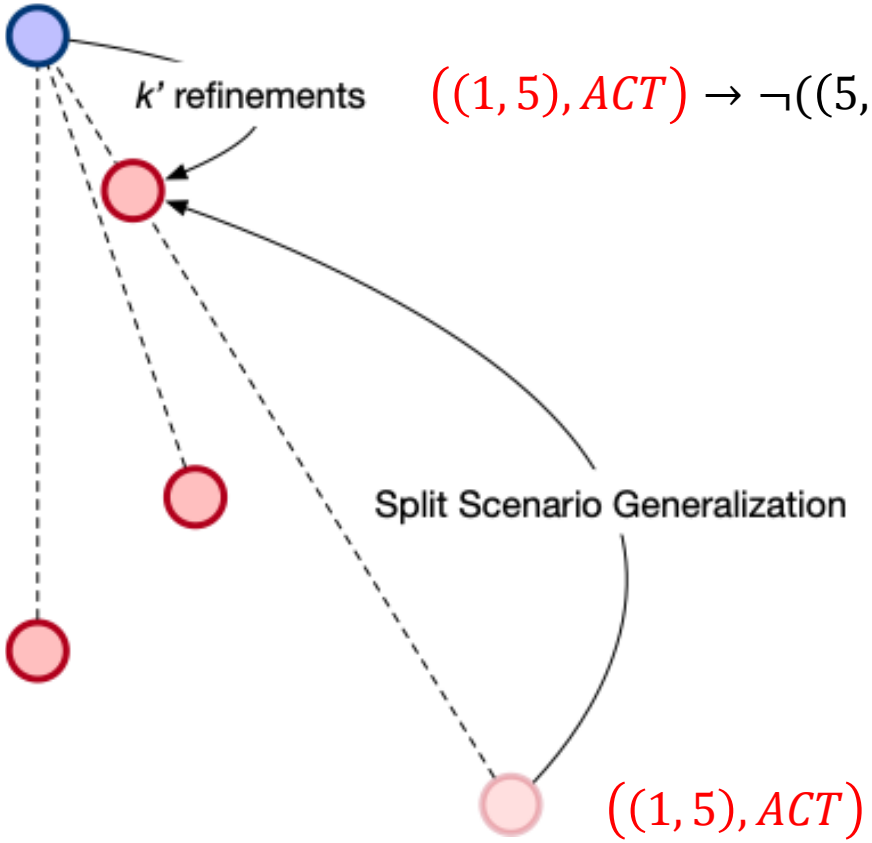
$((1,5), ACT) \wedge ((3,1), INACT) \wedge ((4,1), ACT) \rightarrow \neg((5,7), ACT)$

Enhancing Act. Prop. Usability: Split Scenario Generalization



$$\big((1,5), ACT\big) \wedge \big((3,1), INACT\big) \wedge \big((3,3), INACT\big) \wedge \big((2,1), ACT\big)$$
$$\rightarrow$$
$$\neg((5,7), ACT)$$

Enhancing Act. Prop. Usability: Split Scenario Generalization



$((1,5), ACT) \rightarrow \neg((5,7), ACT)$

Act. Prop. 적용 조건 약화

→

더 적은 refinement/split이 이뤄진 상황에서도 사용 가능!

$((1,5), ACT) \wedge ((3,1), INACT) \wedge ((3,3), INACT) \wedge ((2,1), ACT)$

$\rightarrow$

$\neg((5,7), ACT)$

Act. Prop. DB Supports: Size and Features

- Act. Prop. DB가 지원하는 (구축 및 적용 가능 대상) 심층 신경망 및 요구사항

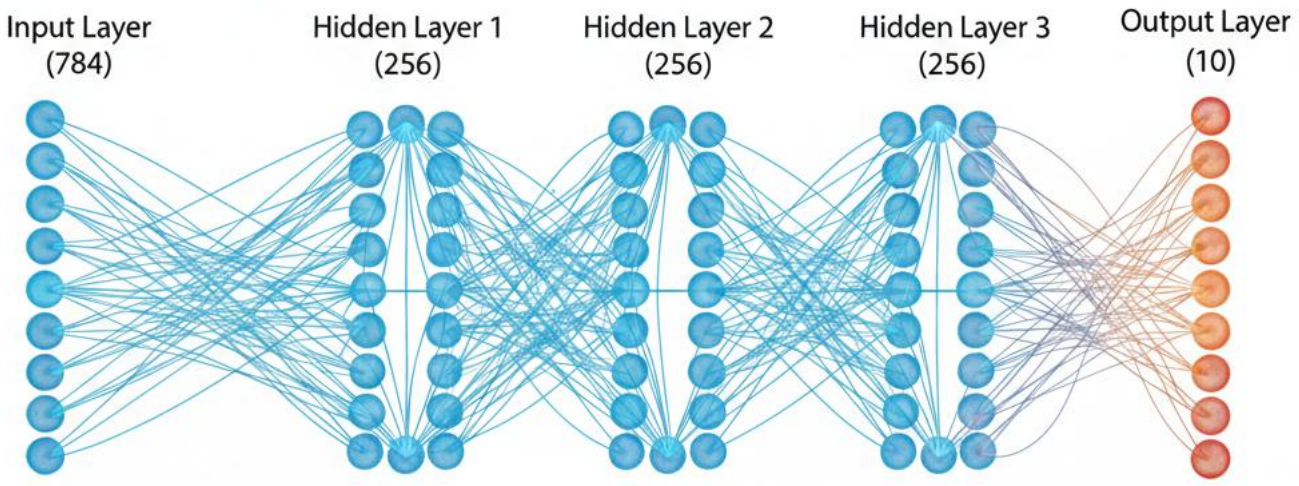
여러 상태로 나눠어 표현될 수 있는 활성화 함수 지원 가능

Architecture	Act. Func.	Dataset/Benchmark	Requirement
FCNN	ReLU, …	AcasXu	Safety
		MNIST	Local Robustness
CNN	ReLU, …	MNIST	Local Robustness
		CIFAR-10 (Oval21)	
		CIFAR-10 (CIFAR-SDP)	

적용 가능 검증 기법: 활성화 함수를 대상으로 추상화 및 refinement 과정을 거치는 기법

Act. Prop. DB Supports: Size and Features

- 실제 Act. Prop. DB 구축 사례 예시
  - Parameters ≅ 400k, 1024개의 ReLU 활성화 함수로 구성되어 있는 MNIST FCNN
  - ~250개*의 MNIST 사진에 대해 local robustness 검증 (Baseline TIMEOUT 제외)



총 1.3만개의 활성화
함수들의 상태 조합 탐색

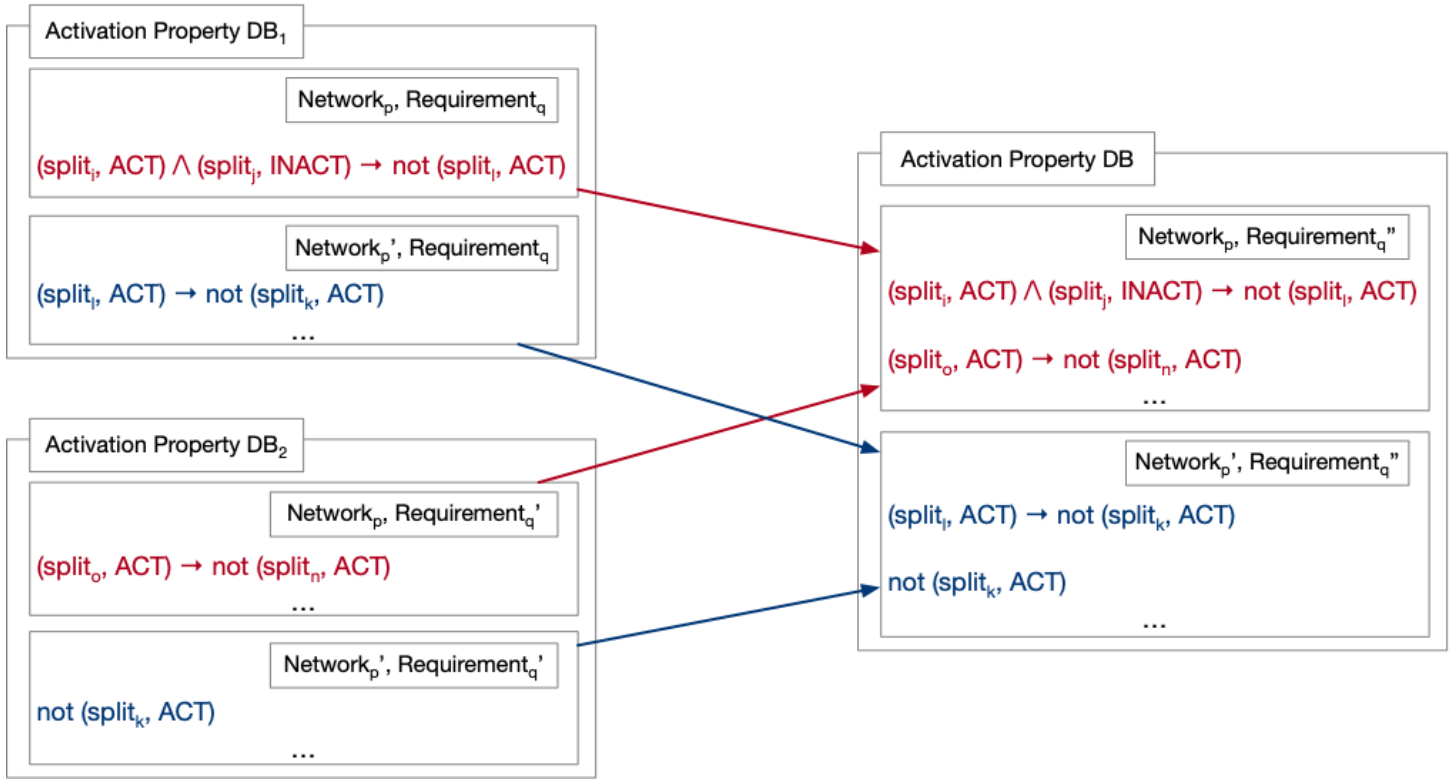
→ ~7300개의 Act. Prop.을 가진 DB 구축 (~5500개의 비일반화 및 ~1800개의 일반화)

Act. Prop. DB Supports: Size and Features

- DB는 Act. Prop.이 생성된 신경망의 구조를 기준으로 분류

Pixels perturbed up to 1%

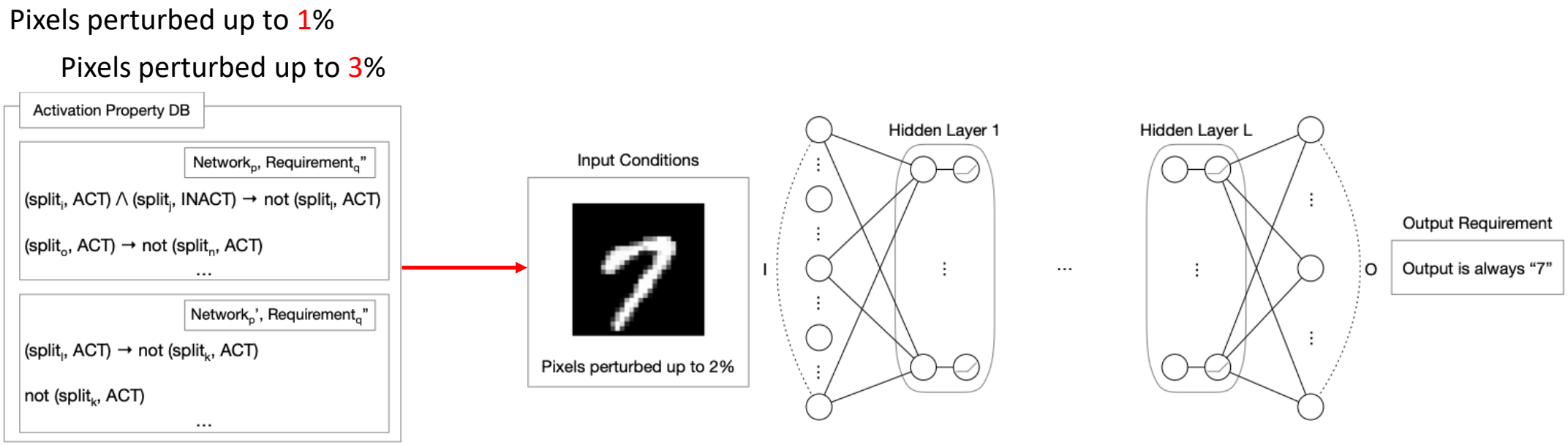
Pixels perturbed up to 3%



→ 여러 검증 과정을 통해 큰 DB 구축 가능

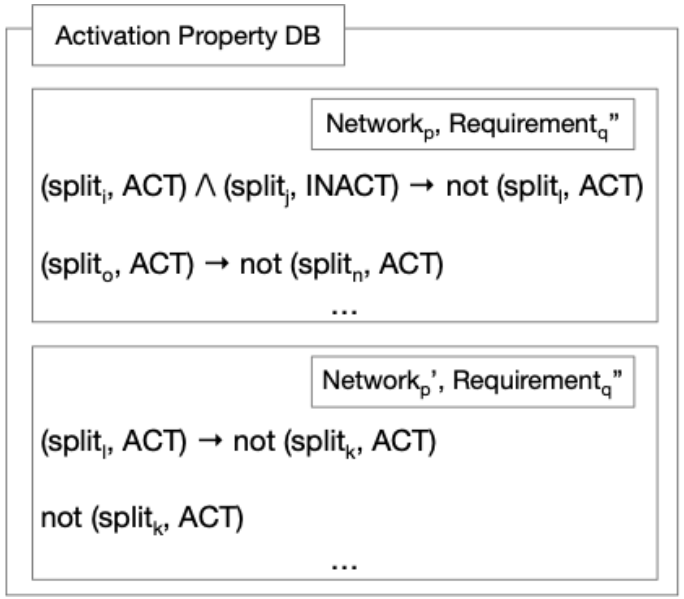
Act. Prop. DB Supports: Size and Features

- DB에 저장된 Act. Prop.은 이를 생성할 때, 검증한 요구사항과 무관하게 적용 가능* (유효할 시)



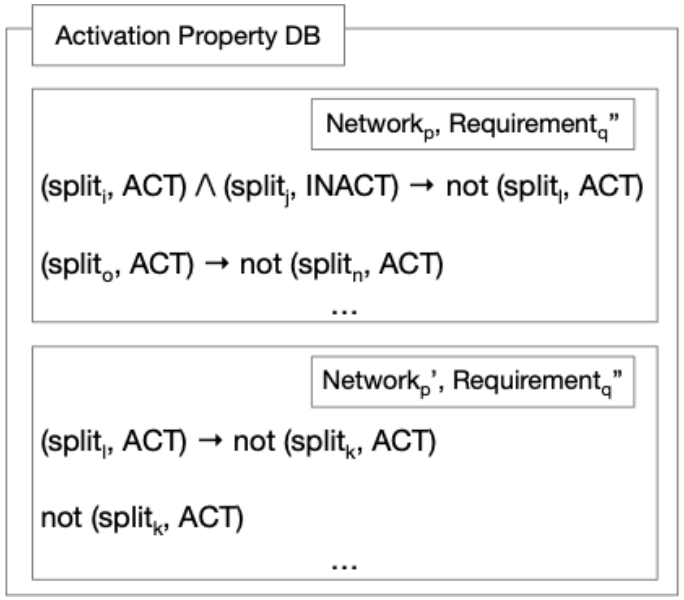
Act. Prop. DB Supports: Size and Features

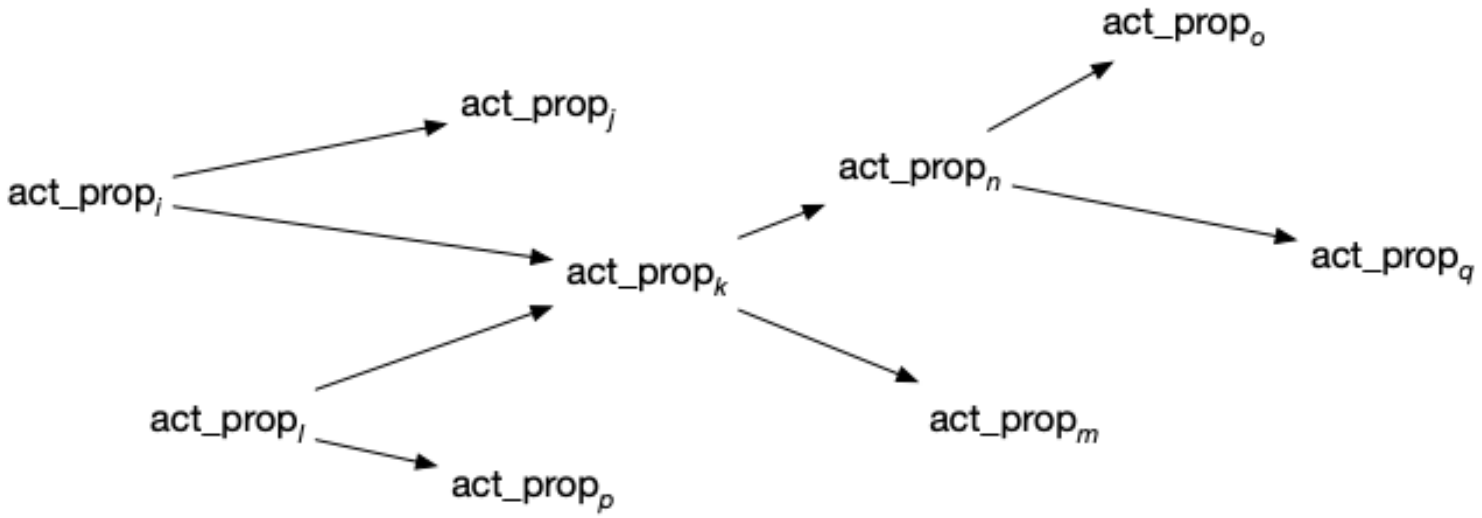
- 효과적인 많은 양의 Act. Prop. 저장 및 로딩을 위해, DAG 구조로 저장



Act. Prop. DB Supports: Size and Features

- 효과적인 많은 양의 Act. Prop. 저장 및 로딩을 위해, DAG 구조로 저장

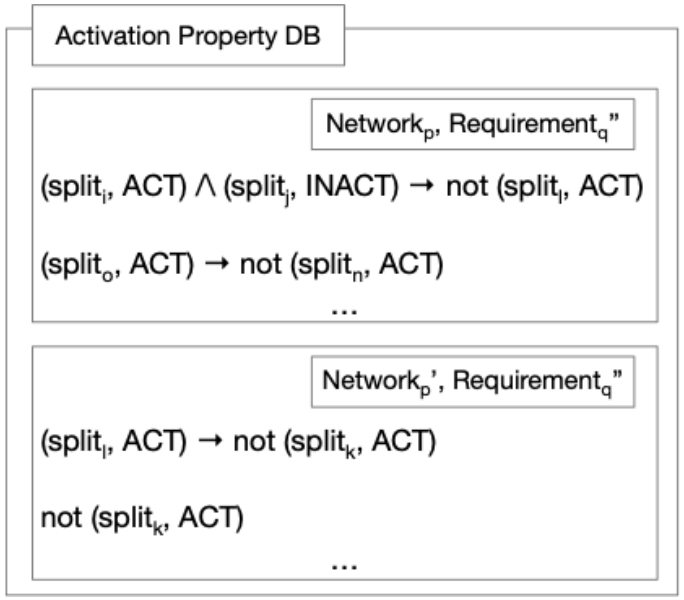


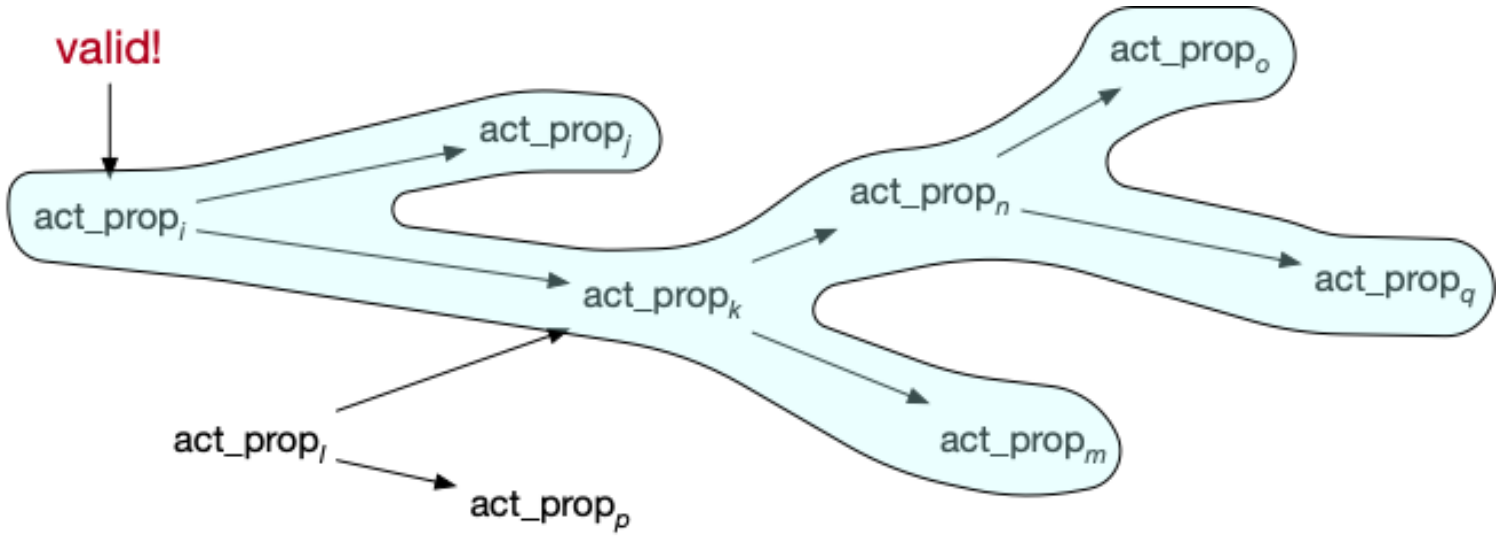


Act. Prop. DB Supports: Size and Features

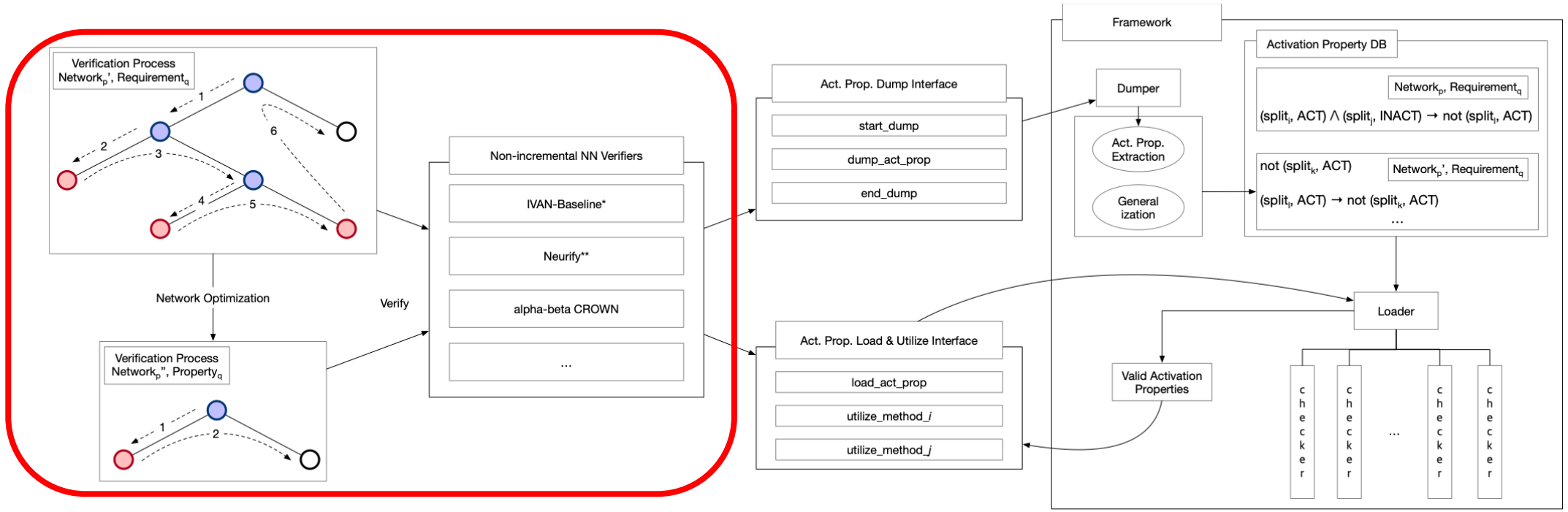
- 효과적인 많은 양의 Act. Prop. 저장 및 로딩을 위해, DAG 구조로 저장

  → 하나의 유효한 Act. Prop. 발견 시, 큰 비용 없이 여러 유효한 Act. Prop. 취득



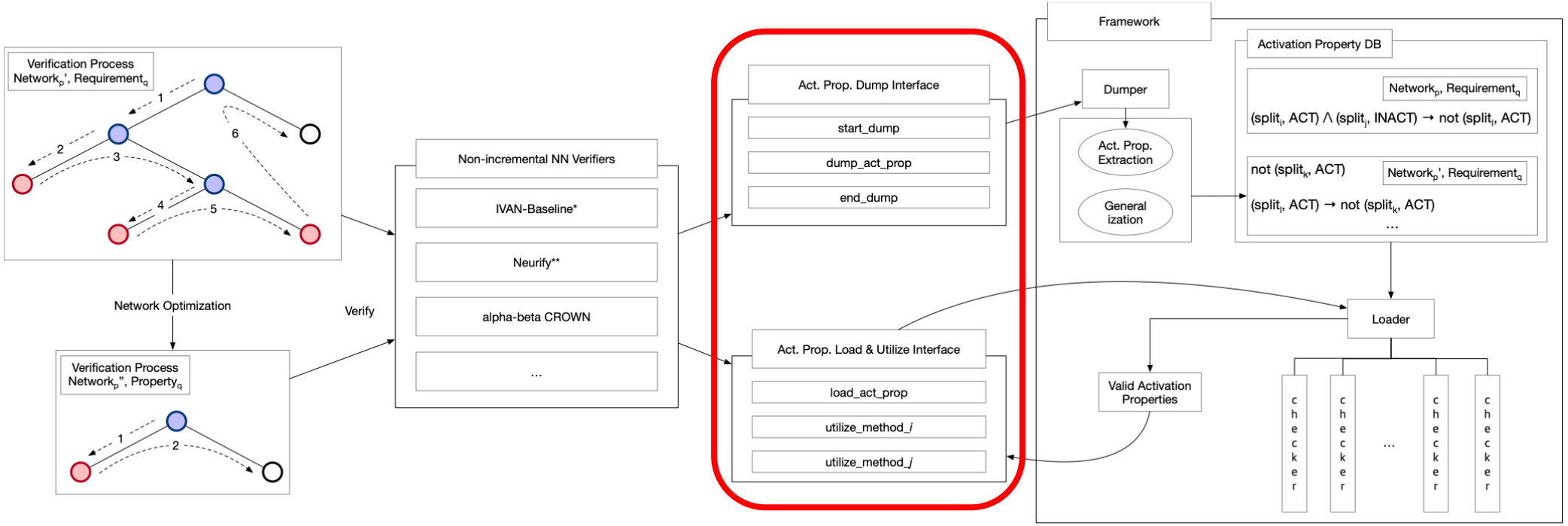


Interface for Non-incremental NN Verifiers



기존 많은 검증기는 반복되는 검증에 최적화되어 있지 않음!

Interface for Non-incremental NN Verifiers

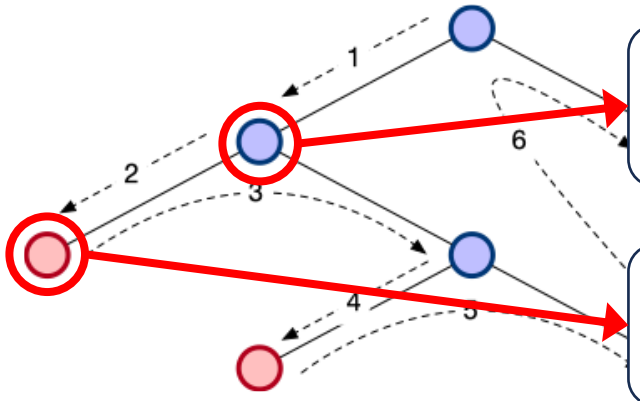


검증기에 큰 수정 없이, 제공되는 인터페이스 호출을 통해 Act. Prop. DB 기반 프레임워크와 연동

Recap: Activation Func. Refinement-based Verification

Algorithm 1 Act. Func. Refinement-based Verif. Algo.

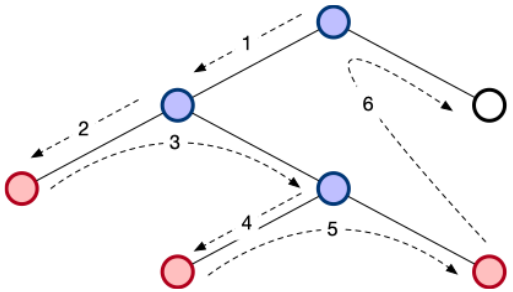
```
1:  state_space ← {initial_state}
2:  while state_space is not empty do
3:      curr_state ← state_space.pop()
4:      (status, cx) ← check_existence(curr_state)
5:      if status = FOUND then
6:          if is_real_counterexample(cx) then
7:              return SAT, cx
8:          else
9:              refined_states ← refine(curr_state)
10:             state_space.add(refined_states)
11:         end if
12:     end if
13: end while
14: return UNSAT
```



Is there a counterexample when
- act. func. at (l_1, n_1) is *ACTIVE* ?

Is there a counterexample when
- act. func. at (l_1, n_1) is *ACTIVE*
- act. func. at (l_2, n_2) is *ACTIVE* ?

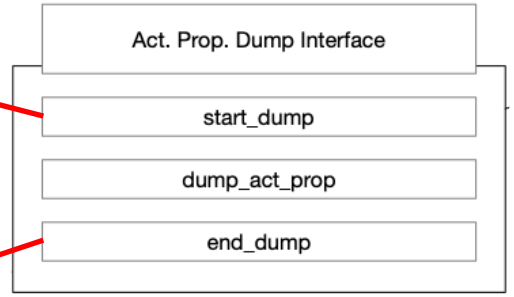
Interface: Accessing Dumping Mechanism



Algorithm 2 Algo. 1 extended with Act. Prop. Generation

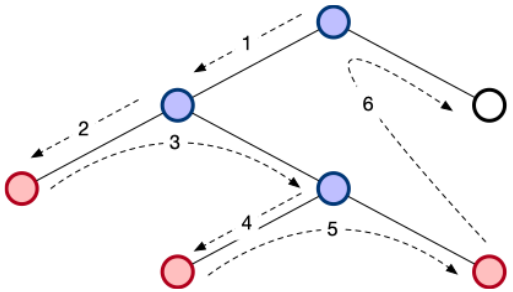
```
1:  state_space ← {initial_state}
2:  dumper.start_dump()
3:  while state_space is not empty do
4:      curr_state ← state_space.pop()
5:      (status, cx) ← check_existence(curr_state)
6:      if status = FOUND then
7:          if is_real_counterexample(cx) then
8:              return SAT, cx
9:          else
10:             refined_states ← refine(curr_state)
11:             state_space.add(refined_states)
12:         end if
13:     else
14:         dumper.dump_act_prop(curr_state)
15:     end if
16: end while
17: dumper.end_dump()
18: return UNSAT
```

검증 과정 시작에 Dump 시작



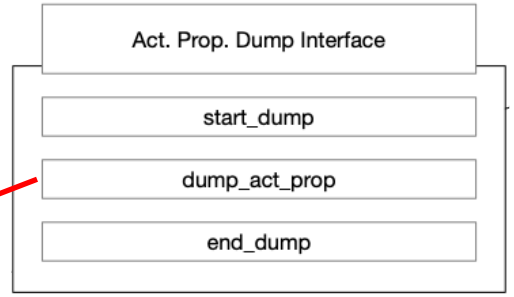
검증 과정 끝에 Dump 종료

Interface: Accessing Dumping Mechanism



Algorithm 2 Algo. 1 extended with Act. Prop. Generation

```
1:  state_space ← {initial_state}
2:  dumper.start_dump()
3:  while state_space is not empty do
4:      curr_state ← state_space.pop()
5:      (status, cx) ← check_existence(curr_state)
6:      if status = FOUND then
7:          if is_real_counterexample(cx) then
8:              return SAT, cx
9:          else
10:             refined_states ← refine(curr_state)
11:             state_space.add(refined_states)
12:         end if
13:     else
14:         dumper.dump_act_prop(curr_state)
15:     end if
16: end while
17: dumper.end_dump()
18: return UNSAT
```



finement 단계마다 Act. Prop. 추출 및 일반화를 진행하는 함수 호출

Experimental Setup

Non-incremental NN Verifier	Benchmark	Applied NN Optimization	Applicable Incremental Verif. Technique
IVAN-Baseline	MNIST	Quantization (Int16, Int8) Pruning (7%, 12%)	IVAN[PLDI 2023]
Neurify*	AcasXu	Random Perturbation (1%)	N\A
$\alpha - \beta$ CROWN	CIFAR-SDP	Random Perturbation (0.5%)	N\A

*: Python 기반 자체 구현 버전

Experimental Setup

RQ: Act. Prop. 기반 점진적 검증 기법은 기존 점진적 검증 기법 대비 효과적인가?

Non-incremental NN Verifier	Benchmark	Applied NN Optimization	Applicable Incremental Verif. Technique
IVAN-Baseline	MNIST	Quantization (Int16, Int8) Pruning (7%, 12%)	IVAN[PLDI 2023]
Neurify*	AcasXu	Random Perturbation (1%)	N\A
$\alpha - \beta$ CROWN	CIFAR-SDP	Random Perturbation (0.5%)	N\A

*: Python 기반 자체 구현 버전

Experimental Setup

RQ: Non-incremental 검증기에 Act. Prop. 기반 점진적 검증 기법을 효과적으로 적용할 수 있을까?

Non-incremental NN Verifier	Benchmark	Applied NN Optimization	Applicable Incremental Verif. Technique
IVAN-Baseline	MNIST	Quantization (Int16, Int8) Pruning (7%, 12%)	IVAN[PLDI 2023]
Neurify*	AcasXu	Random Perturbation (1%)	N\A
$\alpha - \beta$ CROWN	CIFAR-SDP	Random Perturbation (0.5%)	N\A

*: Python 기반 자체 구현 버전

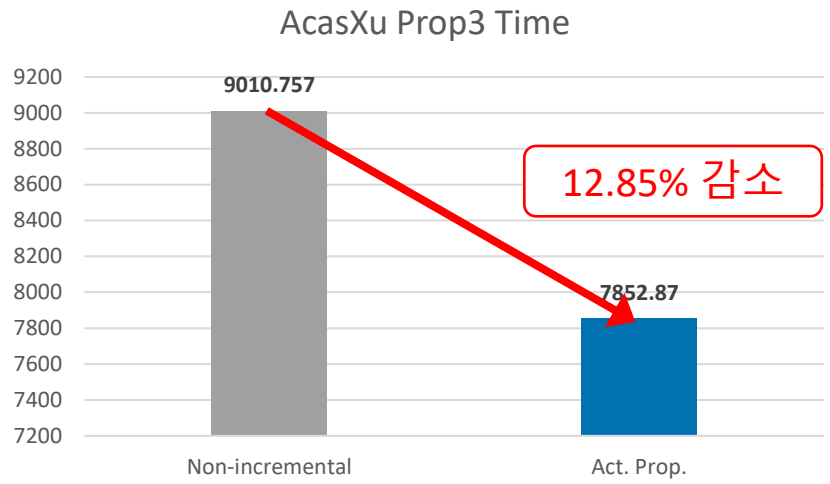
Expanding IVAN-Baseline & Comparison Against IVAN[PLDI 2023]

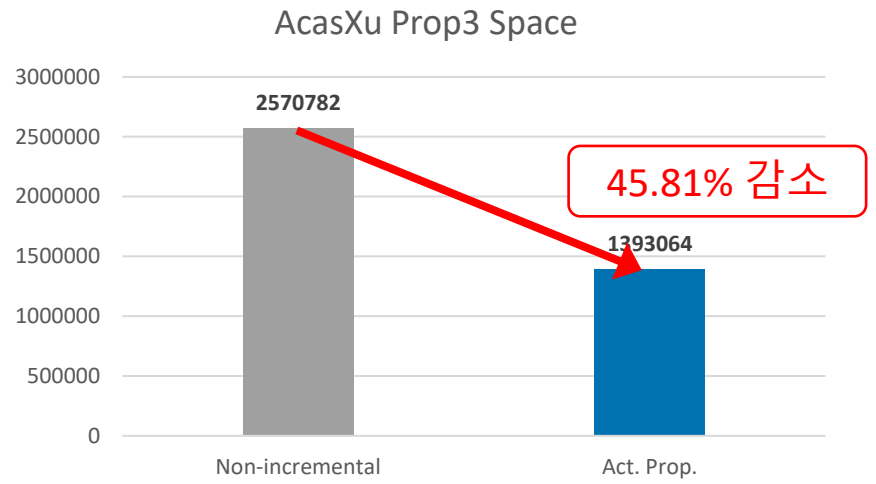
$$time\ speedup = \frac{\sum verification\ time\ (not\ using\ technique)}{\sum verfication\ time\ (when\ using\ technique)}$$

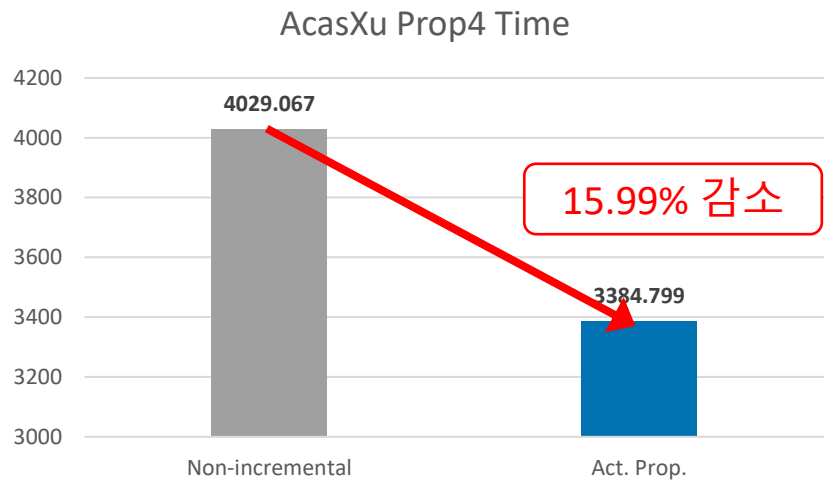
	Optimization	Space Decrease		Time Speedup	
		IVAN	ACT	IVAN	ACT
MNIST L2	Quantization.Int16	1.99	75.9	2.39	2.47
	Quantization.Int8	0.82	2.28	0.96	1.04
	Pruning 7%	1.63	8.21	1.86	1.91
	Pruning 12%	1.49	4.33	1.66	1.75
MNIST L4	Quantization.Int16	1.98	324.71	3.49	2.9
	Quantization.Int8	0.27	4.41	0.52	1.01
	Pruning 7%	1.13	4.84	1.48	1.8
	Pruning 12%	0.78	3.07	1.21	1.42

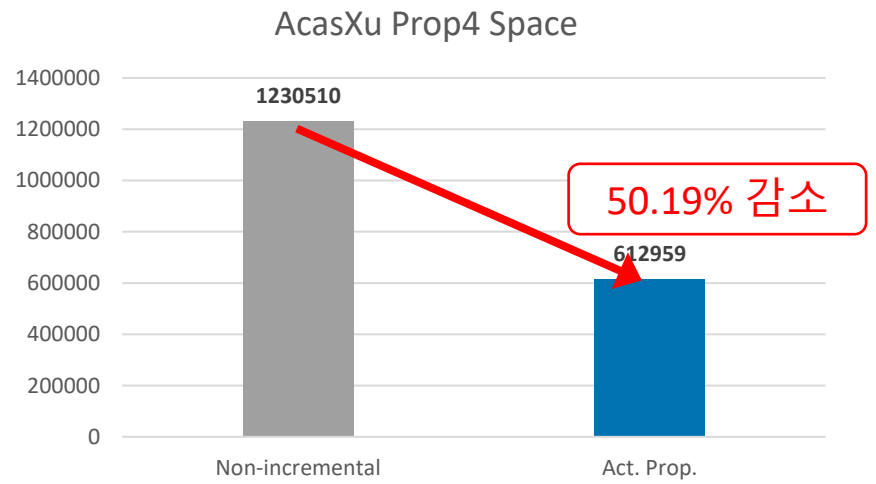
$$space\ decrease = \frac{\sum traversed\ space\ size\ (not\ using\ technique)}{\sum travered\ space\ size\ (when\ using\ technique)}$$

Expanding Neurify*

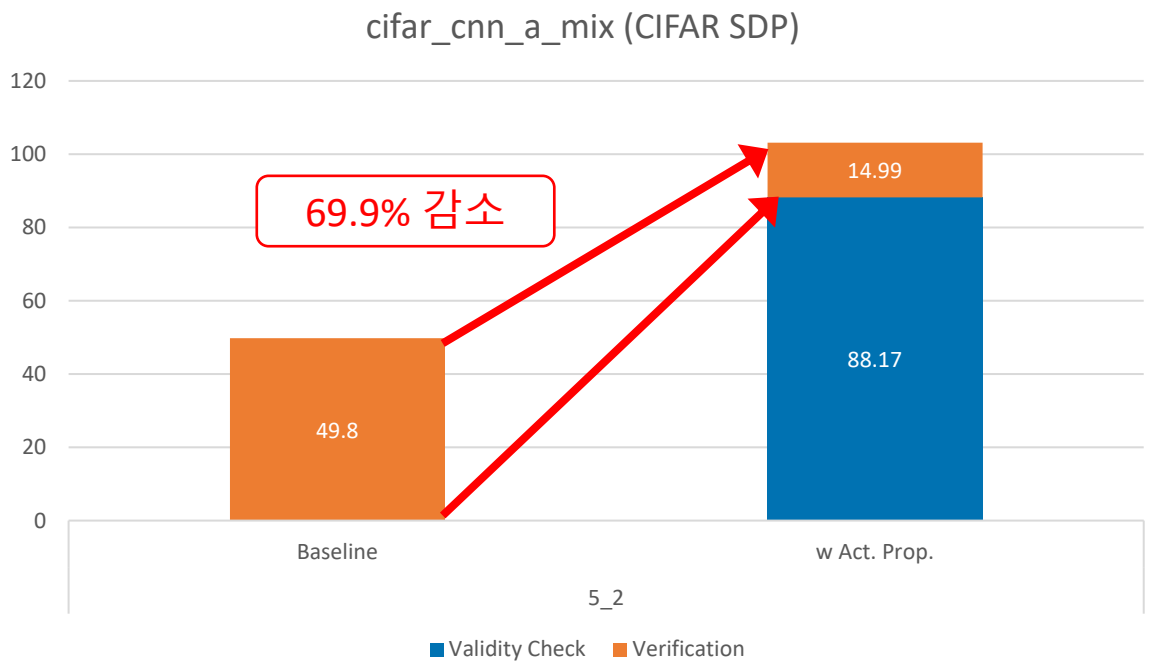


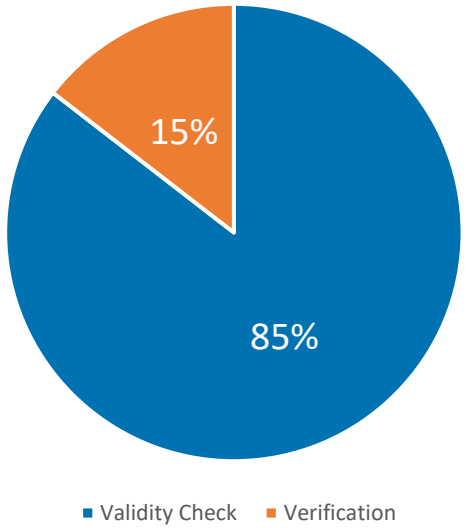






Expanding $\alpha - \beta$ CROWN





Ongoing Work

- 이후 검증 과정에서 진행되는 Act. Prop.의 유효성 검사 비용 최적화
    - 초기 유효성 검사 비용을 여러 검증 과정에 분산시켜, 효과적으로 활용
    - 지원하는 도구의 세팅/구현에 최적화된 유효성 검사 진행 (GPU 활용)

- 연속된 신경망 최적화 과정에, Act. Prop. 적용

  ← 일반화된 Act. Prop.을 더 효과적으로 활용 가능

Thank You

Incremental NN Verification with Implementation-Agnostic Activation Property DB

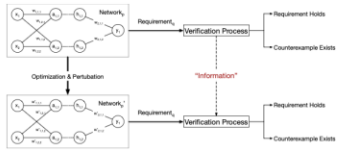
반복된 검증은 더 빠르게: 활성화 성질 기반 심층 신경망 검증 가속

채승현

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

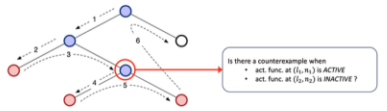
연구 배경

- 반복되는 유사 심층 신경망 검증



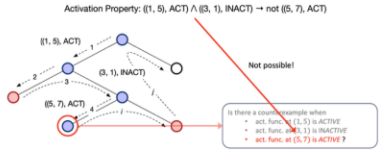
  - 성능 향상을 위한 최적화 과정으로 인해, 동일한 구조와 유사한 매개변수를 갖는 심층 신경망에 대한 검증을 반복하게 됨.
  - 이전 검증 과정의 정보를 활용하여, 이후 검증 효율성을 높이는 연구.

- 활성화 함수 Abstraction and Refinement



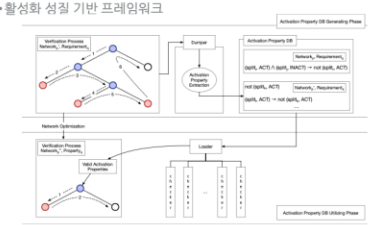
활성화 성질 및 기반 프레임워크

- 활성화 성질 (Activation Property)



  - 복수의 활성화 함수 상태 (INACTIVE, ACTIVE)가 다른 활성화 함수의 상태를 제한하는 성질.
  - 불가능한 문제 공간을 회피하여, 검증 과정에서 탐색해야 하는 문제 공간을 축소함으로써 성능 향상.

- 활성화 성질 기반 프레임워크



  - 활성화 성질 생성 단계
    ⇒ 이전 검증 과정에서 추출한 활성화 성질을 DB로 구축
  - 활성화 성질 활용 단계
    ⇒ 이후 검증 과정에서 이를 로딩하고 활용함으로써 검증 성능 향상

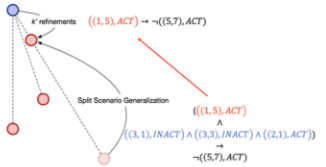
활성화 성질 활용성 증진 기법

- IO 일반화 (Generalization)
  - 활성화 성질은 검증한 요구사항의 입력 조건 하에서만 유효함.

$$(-10 \le x_1 \le 10) \wedge (2 \le x_2 \le 5) \wedge \ldots$$
$$\Rightarrow (-\inf \le x_1 \le \inf) \wedge (2 \le x_2 \le 5) \wedge \ldots$$

  - 유효한 입력 조건을 확장함으로써, 다른 입력 조건을 가진 요구사항 검증 과정에서도 사용 가능.

- Refinement 시나리오 일반화
  - 활성화 성질은 일반적으로 많은 refinement/split 이후에만 발견됨.



  - 활성화 성질 적용 조건을 완화함으로써, 더 적은 refinement/split이 수행된 상황에서도 사용 가능.

비점진적 신경망 검증기를 위한 인터페이스

- 인터페이스 제공

  - 반복되는 검증에 최적화되어 있지 않은 검증기를 점진적 검증에 활용할 수 있도록 확장할 수 있는 인터페이스 제공.
  - 검증기에 큰 수정 없이, 제공되는 API 호출을 통해 활성화 성질 기반 프레임워크와 연동하여 활용 가능.

- 지원 도구
  - Neurify* (Python 기반 자체 구현 버전)
  - IVAN-Baseline
  - alpha-beta CROWN (BICCOS)

실험 결과

- 선행 점진적 검증 기법 IVAN과의 비교

	Optimization	Tree Decrease		Time Speedup	
		IVAN	ACT	IVAN	ACT
MNIST L2	Quantization.Int16	-	-	-	-
	Quantization.Int8	0.82	2.28	0.96	1.04
	Pruning 7%	1.63	8.21	1.86	1.91
	Pruning 12%	1.49	4.33	1.66	1.75
MNIST L4	Quantization.Int16	1.98	324.71	3.49	2.9
	Quantization.Int8	0.27	4.41	0.52	1.01
	Pruning 7%	1.13	4.84	1.48	1.8
	Pruning 12%	0.78	3.07	1.21	1.42

⇒ 활성화 성질 활용에 따른 탐색 공간 축소 및 성능 향상이 관찰됨.

Contact: shchae7@postech.ac.kr