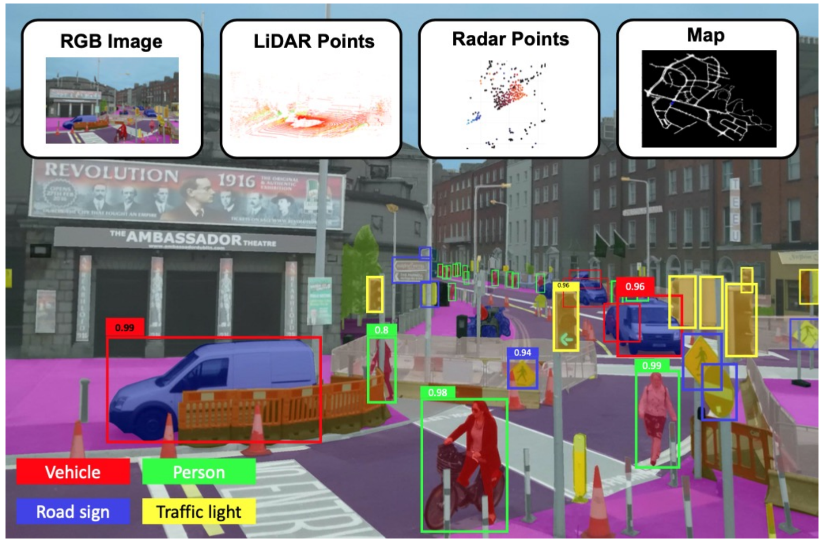
Logical Node Selection Heuristic for Refinement using Spurious Counterexample

컴퓨터공학과 20180462 채승현

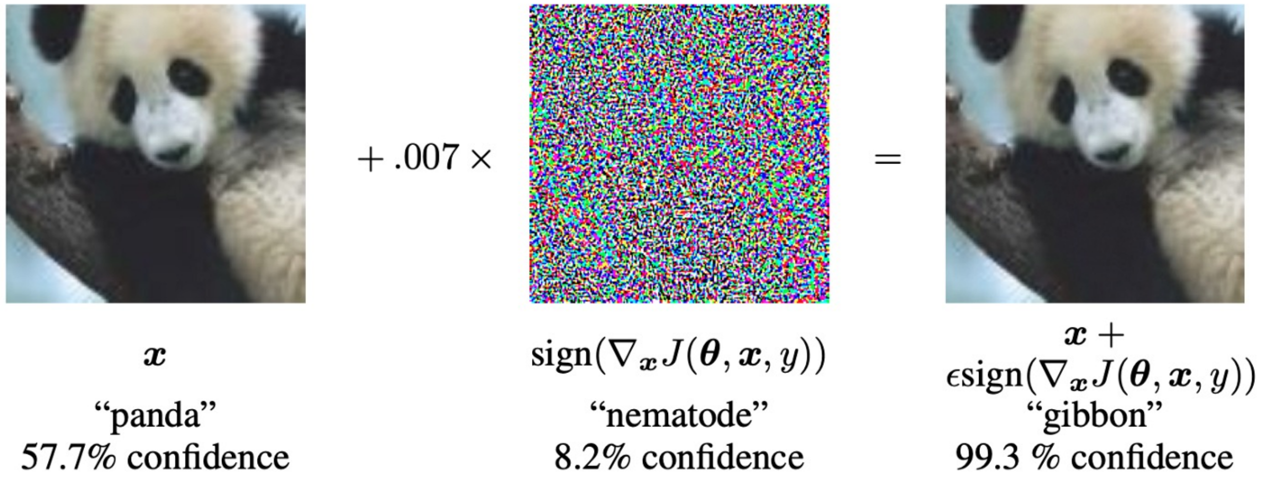
연구 Motivation



Rise in use of DNN (Deep Neural Network)

in safety-critical fields

(e.g. autonomous driving cars)


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

DNN weak against adversarial attacks

→ Need for DNN verification techniques on the rise

연구 목적

Part1

1. Symbolic interval propagation with node splitting refinement method와 branch and bound framework 분석
2. 두 접근 방법의 장점만을 합친 BaB style Neurify 개발

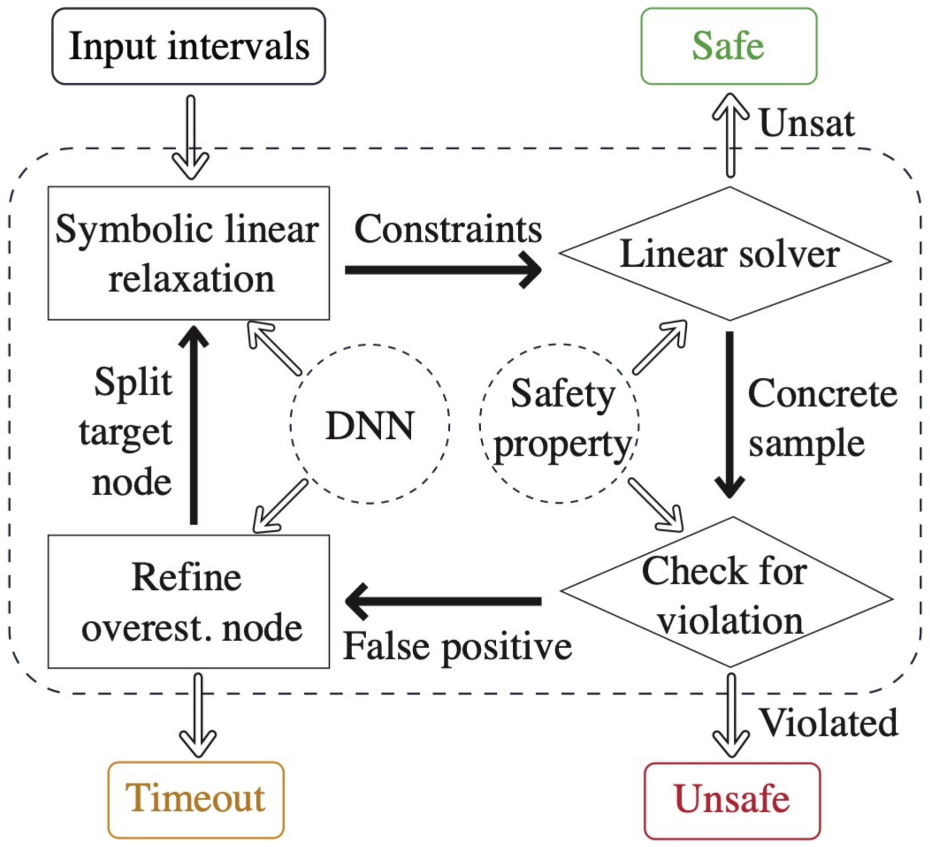
Part2

현재 Node splitting refinement는 전부 경험적 heuristic, 즉 abstraction 과정에서 생성되는 spurious counterexample을 활용하지 못한다는 점을 해결하기 위해

→ spurious counterexample을 활용한 logical node selection heuristic 제시

연구 배경

Symbolic interval propagation with node splitting refinement Overview



Workflow of Neurify

Symbolic interval propagation:
Creating an abstract NN by linearly approximating activation functions, and propagating the input interval to calculate the reachable set

Node splitting refinement:
Based on a certain heuristic, selecting an abstracted node and splitting to refine the abstract NN

연구 배경

Branch and Bound (BaB) Framework Overview

Algorithm 1 Branch and Bound

```
function BaB(net, problem, $\epsilon$)
    global_ub $\leftarrow$ inf
    global_lb $\leftarrow -$inf
    probs $\leftarrow$ [(global_lb, problem)]
    while global_ub $-$ global_lb $> \epsilon$ do
        (_ , prob) $\leftarrow$ pick_out(probs)
        [subprob_1, $\ldots$, subprob_s] $\leftarrow$ split(prob)
        for $i = 1 \ldots s$ do
            prob_ub $\leftarrow$ compute_UB(net, subprob_i)
            prob_lb $\leftarrow$ compute_LB(net, subprob_i)
            if prob_ub $<$ global_ub then
                global_ub $\leftarrow$ prob_ub
                prune_problems(probs, global_ub)
            end if
            if prob_lb $<$ global_ub then
                problems.append((prob_lb, subprob_i))
            end if
        end for
        global_lb $\leftarrow \min\{$lb $\mid$ (lb, prob) $\in$ probs$\}$
    end while
    return global_ub
end function
```

BaB Algorithm

$$\min_{\mathbf{x},\hat{\mathbf{x}}} \quad \hat{x}_n \qquad \text{s.t.} \quad \begin{array}{ll} \mathbf{x}_0 \in \mathcal{C}, & \\ \hat{\mathbf{x}}_{k+1} = W_{k+1}\mathbf{x}_k + \mathbf{b}_{k+1} & k \in [\![0, n-1]\!] \\ \mathbf{x}_k = \sigma_k(\hat{\mathbf{x}}_k) & k \in [\![1, n-1]\!] \end{array}$$

BaB Problem Formulation

Reformulated as a global optimization problem

Repeatedly split input domain into sub-domains each with their own lower/upper bound

Any domain with lower bound > global maximum pruned

연구 방법

(1) Symbolic interval propagation with node splitting refinement method 와 Branch and Bound framework 분석 및 BaB style Neurify 개발
   (1) 대표적인 도구 Neurify, BaB를 활용하여 Acas Xu 데이터셋 NN 검증
   (2) 알고리즘 및 검증 결과 분석을 통한 장단점 정리
   (3) 두 접근의 장점들을 합친 BaB style Neurify 개발 (BaBNeurify)

(2) Node splitting refinement method에서 spurious counterexample을 활용한 logical node selection heuristic 제시
   (1) NN 검증에서 spurious counterexample의 의미 분석
   (2) Spurious counterexample이 미치는 영향 측정 방법 고안
   (3) Counterexample을 활용한 새로운 heuristic 제시

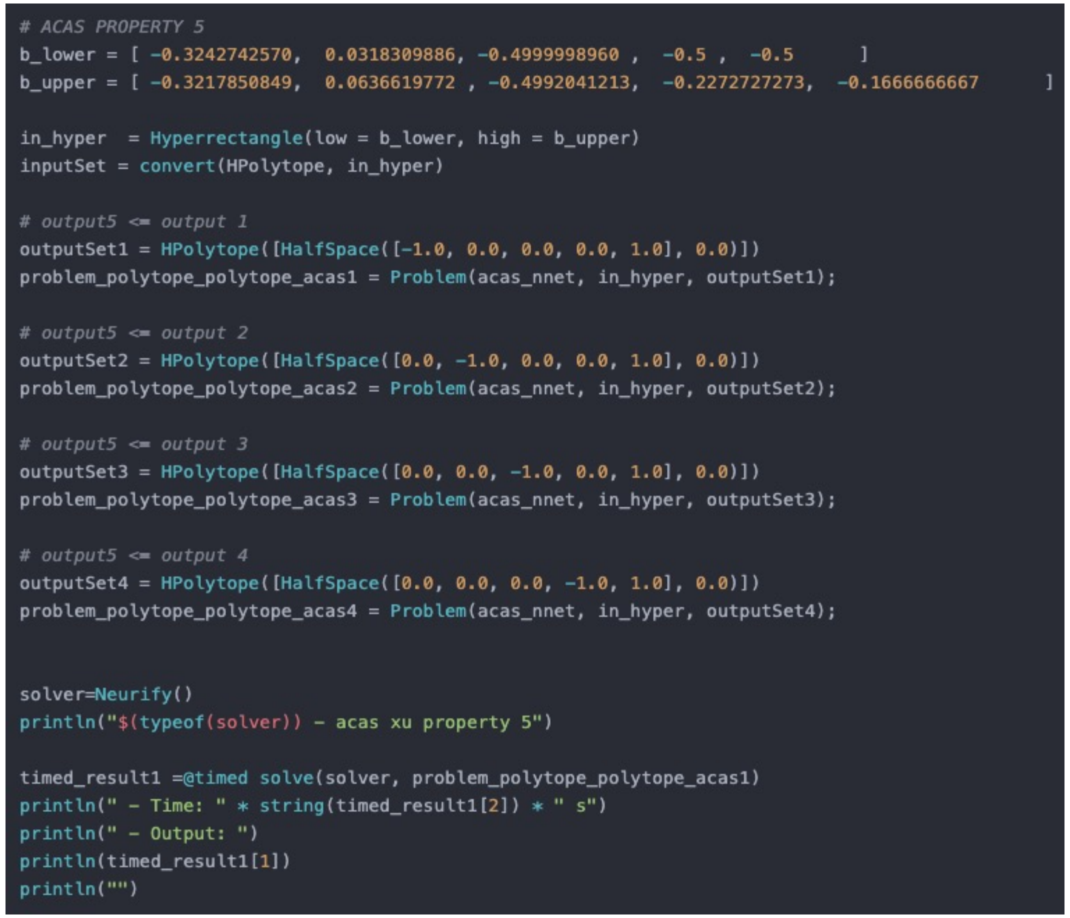
연구 결과 및 평가

1.1 Neurify와 BaB 도구를 통한 Acas Xu 데이터셋 검증 - 1

```julia
function solve(solver::BaB, problem::Problem)
    (u_approx, u, x_u) = output_bound(solver, problem, :max)
    (l_approx, l, x_l) = output_bound(solver, problem, :min)
    bound = Hyperrectangle(low = [l], high = [u])
    reach = Hyperrectangle(low = [l_approx], high = [u_approx])

    output = problem.output
    if reach ⊆ output
        return ReachabilityResult(:holds, [reach])
    end
    high(bound) > high(output) && return CounterExampleResult(:violated, x_u)
    low(bound)  < low(output)  && return CounterExampleResult(:violated, x_l)
    return ReachabilityResult(:unknown, [reach])
end
```

BaB Julia Code

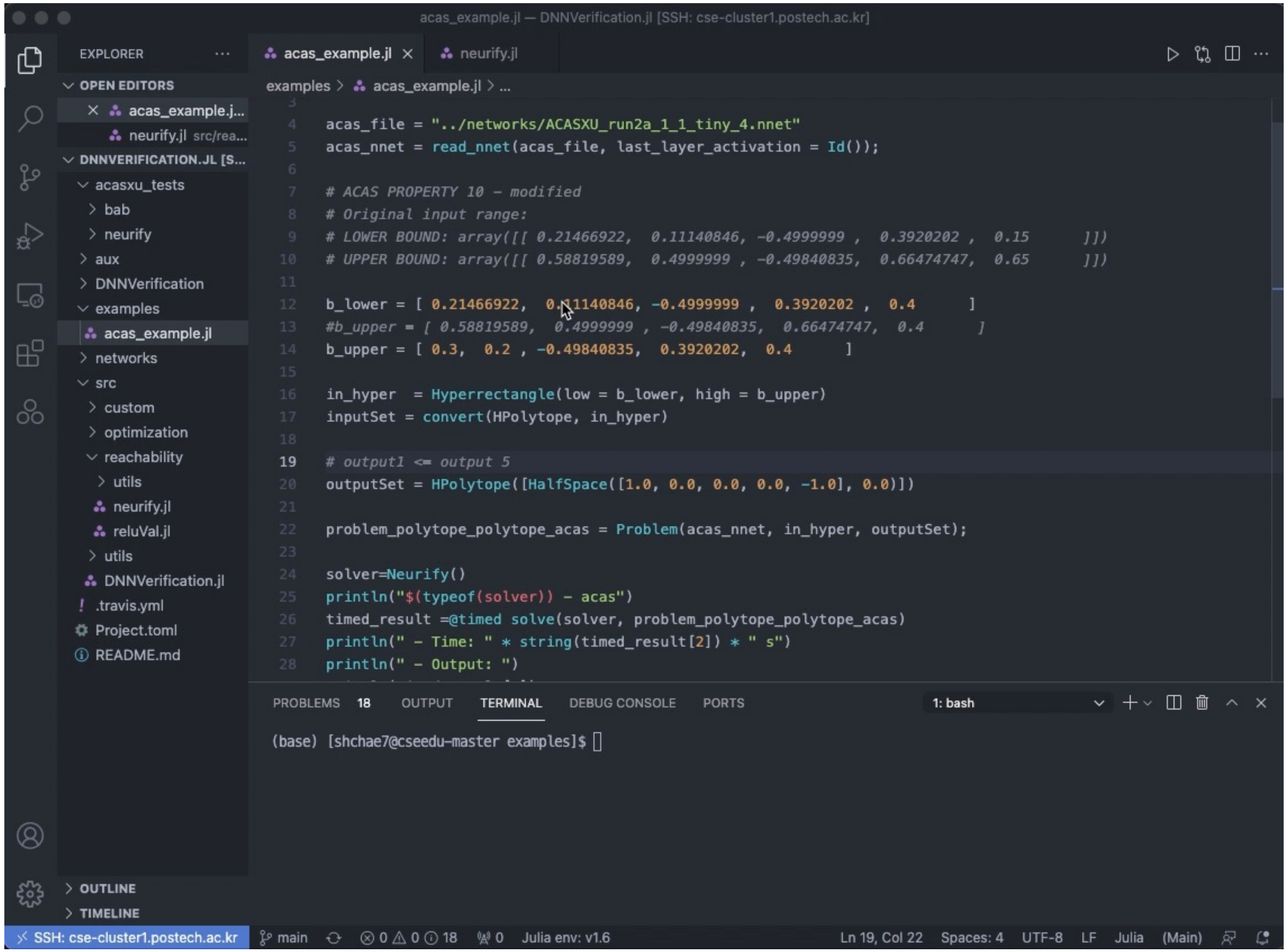
https://github.com/shchae7/DNNVerification.jl



```julia
# ACAS PROPERTY 5
b_lower = [ -0.3242742570,  0.0318309886, -0.4999998960 ,  -0.5 ,  -0.5      ]
b_upper = [ -0.3217850849,  0.0636619772 , -0.4992041213,  -0.2272727273,  -0.1666666667      ]

in_hyper  = Hyperrectangle(low = b_lower, high = b_upper)
inputSet = convert(HPolytope, in_hyper)

# output5 <= output 1
outputSet1 = HPolytope([HalfSpace([-1.0, 0.0, 0.0, 0.0, 1.0], 0.0)])
problem_polytope_polytope_acas1 = Problem(acas_nnet, in_hyper, outputSet1);

# output5 <= output 2
outputSet2 = HPolytope([HalfSpace([0.0, -1.0, 0.0, 0.0, 1.0], 0.0)])
problem_polytope_polytope_acas2 = Problem(acas_nnet, in_hyper, outputSet2);

# output5 <= output 3
outputSet3 = HPolytope([HalfSpace([0.0, 0.0, -1.0, 0.0, 1.0], 0.0)])
problem_polytope_polytope_acas3 = Problem(acas_nnet, in_hyper, outputSet3);

# output5 <= output 4
outputSet4 = HPolytope([HalfSpace([0.0, 0.0, 0.0, -1.0, 1.0], 0.0)])
problem_polytope_polytope_acas4 = Problem(acas_nnet, in_hyper, outputSet4);


solver=Neurify()
println("$(typeof(solver)) - acas xu property 5")

timed_result1 =@timed solve(solver, problem_polytope_polytope_acas1)
println(" - Time: " * string(timed_result1[2]) * " s")
println(" - Output: ")
println(timed_result1[1])
println("")
```

Neurify Property Check Code

연구 결과 및 평가



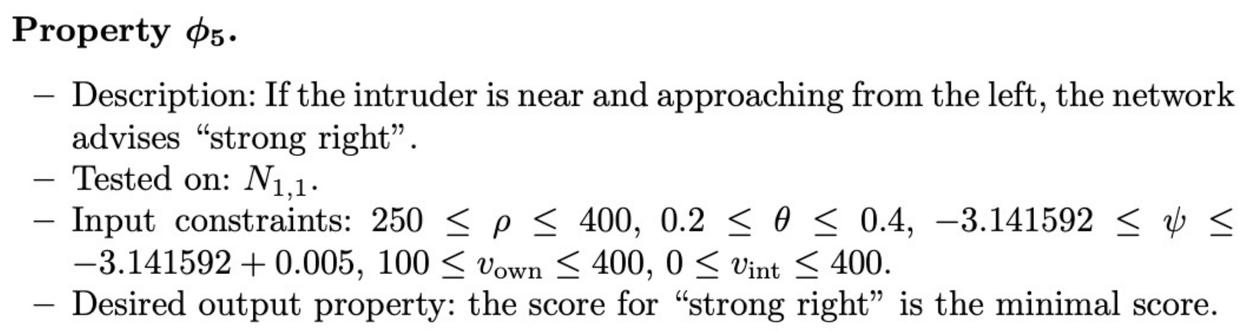
acas_example.jl — DNNVerification.jl [SSH: cse-cluster1.postech.ac.kr]

```julia
acas_file = "../networks/ACASXU_run2a_1_1_tiny_4.nnet"
acas_nnet = read_nnet(acas_file, last_layer_activation = Id());

# ACAS PROPERTY 10 - modified
# Original input range:
# LOWER BOUND: array([[ 0.21466922,  0.11140846, -0.4999999 ,  0.3920202 ,  0.15      ]])
# UPPER BOUND: array([[ 0.58819589,  0.4999999 , -0.49840835,  0.66474747,  0.65      ]])

b_lower = [ 0.21466922,  0.11140846, -0.4999999 ,  0.3920202 ,  0.4      ]
#b_upper = [ 0.58819589,  0.4999999 , -0.49840835,  0.66474747,  0.4      ]
b_upper = [ 0.3,  0.2 , -0.49840835,  0.3920202,  0.4      ]

in_hyper  = Hyperrectangle(low = b_lower, high = b_upper)
inputSet = convert(HPolytope, in_hyper)

# output1 <= output 5
outputSet = HPolytope([HalfSpace([1.0, 0.0, 0.0, 0.0, -1.0], 0.0)])

problem_polytope_polytope_acas = Problem(acas_nnet, in_hyper, outputSet);

solver=Neurify()
println("$(typeof(solver)) - acas")
timed_result =@timed solve(solver, problem_polytope_polytope_acas)
println(" - Time: " * string(timed_result[2]) * " s")
println(" - Output: ")
```

```
(base) [shchae7@cseedu-master examples]$ 
```

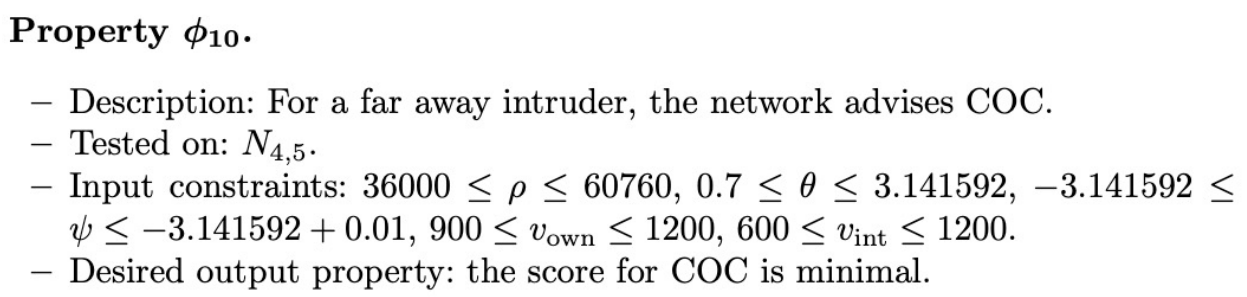
연구 결과 및 평가

1.1 Neurify와 BaB 도구를 통한 Acas Xu 데이터셋 검증 - 2



Property ϕ_5.

- Description: If the intruder is near and approaching from the left, the network advises "strong right".
- Tested on: $N_{1,1}$.
- Input constraints: $250 \leq \rho \leq 400$, $0.2 \leq \theta \leq 0.4$, $-3.141592 \leq \psi \leq -3.141592 + 0.005$, $100 \leq v_{\text{own}} \leq 400$, $0 \leq v_{\text{int}} \leq 400$.
- Desired output property: the score for "strong right" is the minimal score.



Property ϕ_{10}.

- Description: For a far away intruder, the network advises COC.
- Tested on: $N_{4,5}$.
- Input constraints: $36000 \leq \rho \leq 60760$, $0.7 \leq \theta \leq 3.141592$, $-3.141592 \leq \psi \leq -3.141592 + 0.01$, $900 \leq v_{\text{own}} \leq 1200$, $600 \leq v_{\text{int}} \leq 1200$.
- Desired output property: the score for COC is minimal.

	Neurify	BaB
Prop 5	140.3601s	TIMEOUT
Prop 10	92.593s	TIMEOUT

TIMEOUT: 12h

연구 결과 및 평가

1.2 알고리즘 및 검증 결과 분석을 통한 장단점 정리

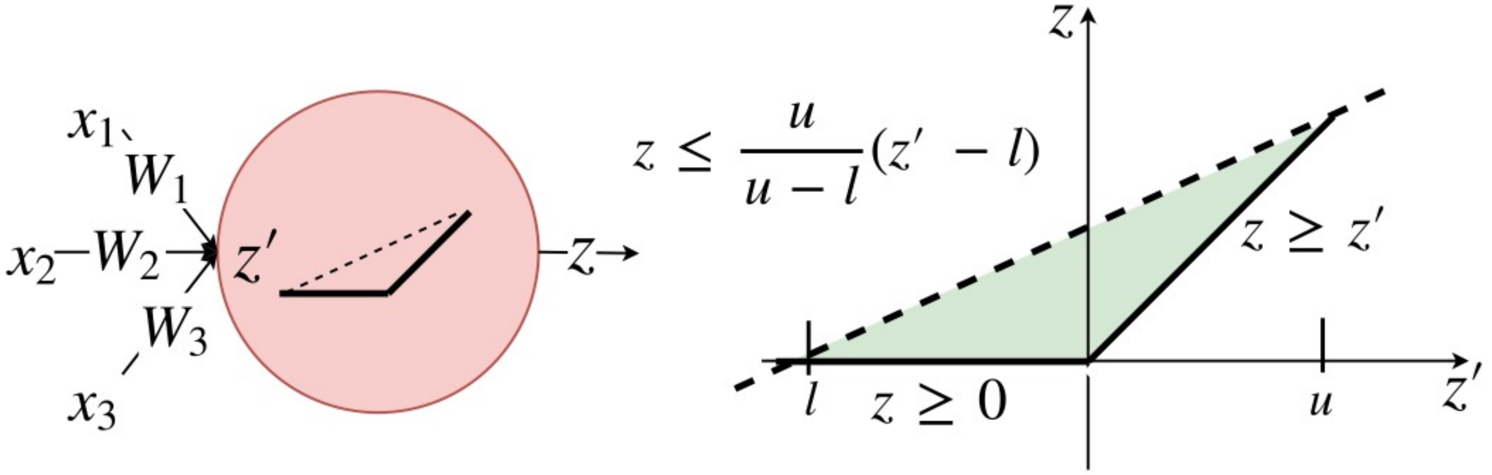
	Neurify	BaB
Pros	경험적 heuristic (max gradient를 가진 node 선택)을 사용 현재 high-dimension input 상대 state-of-the art	Global optimization problem으로 formulate함으로써 node selection heuristic에서 splitting decision이 output에 영향을 미치는 정도를 측정하는데 용이
Cons	Abstraction으로 인해 발생하는 Spurious counterexample 미사용	Naïve한 heuristic 사용

$$\min_{\mathbf{x},\hat{\mathbf{x}}} \quad \hat{x}_n \qquad \text{s.t.} \quad \begin{aligned} &\mathbf{x}_0 \in \mathcal{C}, && \\ &\hat{\mathbf{x}}_{k+1} = W_{k+1}\mathbf{x}_k + \mathbf{b}_{k+1} && k \in [\![0, n-1]\!] \\ &\mathbf{x}_k = \sigma_k\left(\hat{\mathbf{x}}_k\right) && k \in [\![1, n-1]\!] \end{aligned}$$

BaB style Neurify (BaBNeurify) 개발

연구 결과 및 평가

Neural network verification에서 spurious counterexample이란



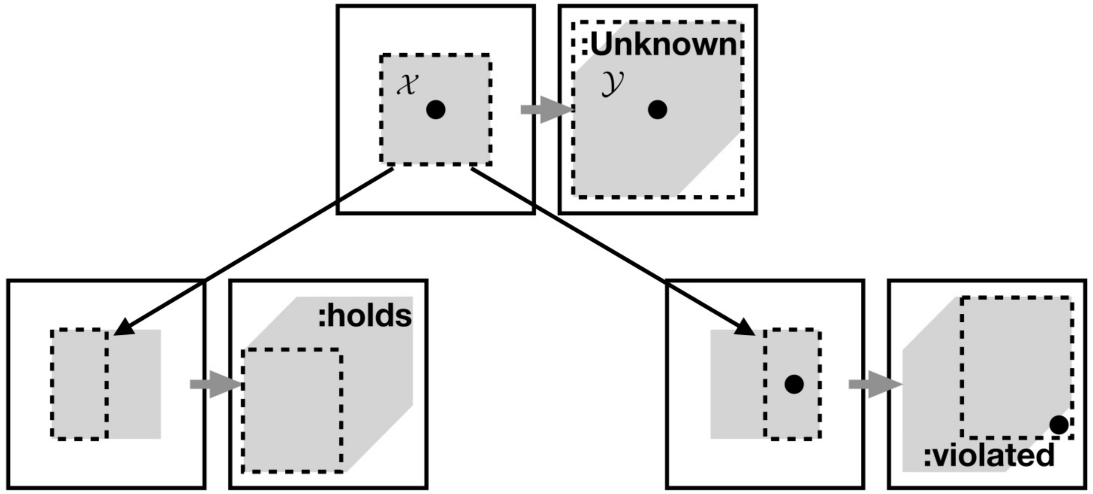
Linear relaxation of a ReLU node

Non-linear activation function의 linear approximation을 위한 abstraction에서의 coarseness로 인해 발생하는 false positive counterexample (abstract NN에서만 counterexample)

→ Concrete NN과 abstract NN의 행동 차이의 결과물

연구 결과 및 평가

Counterexample 이 미치는 영향 측정 방법 및 counterexample을 활용한 새로운 heuristic 고안



Node splitting process/refinement

Domain의 lower bound가 0보다 작을 때

(property를 성립하지 않는 경우)

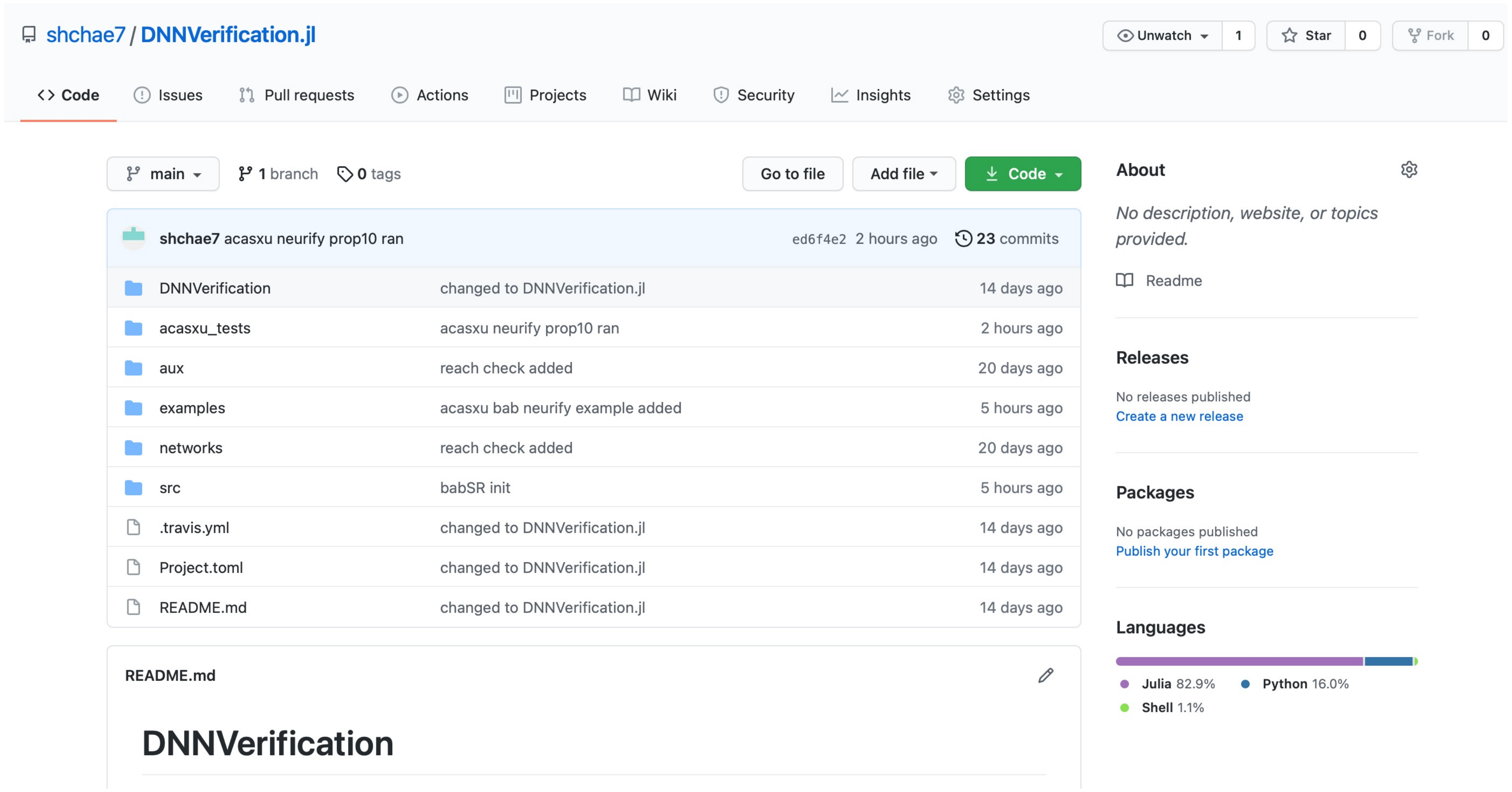
1. Split 후 subdomain의 lower bound와 0 사이 거리
2. Split 후, original domain과 subdomain의 lower bound 차이

Heuristic

양쪽 모두 spurious counterexample이 존재하는 node들을 전부 확인한 후 옆 측정 방법 중 하나를 사용하여 가장 크게 영향을 미치는 node 선택

연구 결과 및 평가

1.3 두 접근의 장점들을 합친 BaB style Neurify 개발 (BaBNeurify)



shchae7 / DNNVerification.jl

shchae7 acasxu neurify prop10 ran ed6f4e2 2 hours ago 23 commits

DNNVerification	changed to DNNVerification.jl	14 days ago
acasxu_tests	acasxu neurify prop10 ran	2 hours ago
aux	reach check added	20 days ago
examples	acasxu bab neurify example added	5 hours ago
networks	reach check added	20 days ago
src	babSR init	5 hours ago
.travis.yml	changed to DNNVerification.jl	14 days ago
Project.toml	changed to DNNVerification.jl	14 days ago
README.md	changed to DNNVerification.jl	14 days ago

README.md

DNNVerification

About

No description, website, or topics provided.

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Julia 82.9% ● Python 16.0%
● Shell 1.1%

토론 및 전망

Abstraction 이로 인해 필히 생길 수 밖에 없는 현재 효과적으로 활용되고 있지 않았던 spurious counterexample을 활용하는 heuristic 제시하기 시작

여러 heuristic을 modular하게 교체할 수 있는 framework를 개발하는 과정으로, 앞으로 새로 고안되는 heuristic 손쉽게 확인할 수 있는 플랫폼을 제공할 수 있을 것이라고 기대

SAT 혹은 SMT solver에서 활용하는 conflict driven learning과 같이 counterexample을 활용한 heuristic을 통해 검증 성능 향상을 얻을 수 있다고 기대

감사합니다