```
Last login: Thu Jul 29 14:15:41 on ttys000
Chioma_N@Akwarandus-MacBook-Air ~ % exec '/Applications/Julia-1.6.app/Contents/R
esources/julia/bin/julia'
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.6.1 (2021-04-23)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using NeuralVerification, LazySets


julia> import NeuralVerification: ReLU, Id

julia> lowerInput = [10.0, 35.0, 0.0,
              10.0, 35.0, 6.0,
              10.0, 35.0, 16.0,
              10.0, 35.0, 22.0,
              10.0, 35.0, 24.0,
              10.0, 35.0, 30.0,
              10.0, 35.0, 36.0,
              10.0, 35.0, 42.0,
              10.0, 35.0, 52.0,
              10.0, 35.0, 58.0]
30-element Vector{Float64}:

 10.0
 35.0
  0.0
 10.0
 35.0
  6.0
 10.0
 35.0
 16.0
 10.0
  ⋮
 10.0
 35.0
 42.0
 10.0
 35.0
 52.0
 10.0
 35.0
 58.0

julia> upperInput = [10.0, 35.0, 5.0,
              10.0, 35.0, 11.0,
              10.0, 35.0, 21.0,
              10.0, 35.0, 23.0,
              10.0, 35.0, 29.0,
              10.0, 35.0, 35.0,
              10.0, 35.0, 41.0,
              10.0, 35.0, 47.0,
              10.0, 35.0, 57.0,
              10.0, 35.0, 63.0]
30-element Vector{Float64}:
 10.0
 35.0
  5.0
 10.0
 35.0
 11.0
 10.0
 35.0
 21.0
 10.0
  ⋮
 10.0
 35.0
 47.0
 10.0
 35.0
 57.0
 10.0
 35.0
 63.0

julia> x = Hyperrectangle(low = lowerInput, high = upperInput)
Hyperrectangle{Float64, Vector{Float64}, Vector{Float64}}([10.0, 35.0, 2.5, 10.0
, 35.0, 8.5, 10.0, 35.0, 18.5, 10.0  …  38.5, 10.0, 35.0, 44.5, 10.0, 35.0, 54.5
, 10.0, 35.0, 60.5], [0.0, 0.0, 2.5, 0.0, 0.0, 2.5, 0.0, 0.0, 2.5, 0.0  …  2.5,
0.0, 0.0, 2.5, 0.0, 0.0, 2.5, 0.0, 0.0, 2.5])

julia> y = Hyperrectangle(low = [-1000], high = [-0.001])
Hyperrectangle{Float64, Vector{Float64}, Vector{Float64}}([-500.0005], [499.9995
])

julia> model = "/Users/Chioma_N/Desktop/object/PCC-RL-master/src/gym/tensorflow/
reLuNet/resNet"
"/Users/Chioma_N/Desktop/object/PCC-RL-master/src/gym/tensorflow/reLuNet/resNet"

julia> net = read_nnet(model)

Network(NeuralVerification.Layer[NeuralVerification.Layer{ReLU, Float64}([-0.476
940215 -0.233635619 … 0.260245979 0.0854543895; -0.279218972 0.212435156 … -0.30
4853797 -0.287973344; … ; 0.043456167 -0.299254805 … -0.336141109 0.0869233087;
0.0609097183 -0.0746256411 … -0.356662869 -0.211450607], [-0.03726024, 0.0, 0.0,
 0.06048633, -0.11259795, 0.0, 0.0, -0.07211282, 0.044972, 0.0, 0.0, 0.0, 0.0, 0
.01267096, -0.00579271, 0.0], ReLU()), NeuralVerification.Layer{ReLU, Float64}([
-0.02278417 -0.15391123 … -0.40684664 -0.21582426; 0.27223375 0.23778197 … 0.137
33128 0.12970361; … ; 0.18829295 0.33743098 … 0.06836078 0.16406111; 0.37219688
-0.16067436 … -0.24720219 0.12515154], [0.0, 0.0, 0.06707639, 0.0, -0.07552656,
0.00930779, -0.00740244, 0.0, -0.04904214, 0.0, -0.02598077, 0.03745193, 0.03417
419, -0.01520709, 0.0, 0.01063646], ReLU()), NeuralVerification.Layer{Id, Float6
4}([0.30263627 -0.23838618 … -0.09799442 -0.00773645], [0.04735997], Id())])

julia> prob = Problem(net, x, y)
Problem{Hyperrectangle{Float64, Vector{Float64}, Vector{Float64}}, Hyperrectangl
e{Float64, Vector{Float64}, Vector{Float64}}}(Network(NeuralVerification.Layer[N
euralVerification.Layer{ReLU, Float64}([-0.476940215 -0.233635619 … 0.260245979
0.0854543895; -0.279218972 0.212435156 … -0.304853797 -0.287973344; … ; 0.043456
167 -0.299254805 … -0.336141109 0.0869233087; 0.0609097183 -0.0746256411 … -0.35
6662869 -0.211450607], [-0.03726024, 0.0, 0.0, 0.06048633, -0.11259795, 0.0, 0.0
, -0.07211282, 0.044972, 0.0, 0.0, 0.0, 0.0, 0.01267096, -0.00579271, 0.0], ReLU
()), NeuralVerification.Layer{ReLU, Float64}([-0.02278417 -0.15391123 … -0.40684
664 -0.21582426; 0.27223375 0.23778197 … 0.13733128 0.12970361; … ; 0.18829295 0
.33743098 … 0.06836078 0.16406111; 0.37219688 -0.16067436 … -0.24720219 0.125151
54], [0.0, 0.0, 0.06707639, 0.0, -0.07552656, 0.00930779, -0.00740244, 0.0, -0.0
4904214, 0.0, -0.02598077, 0.03745193, 0.03417419, -0.01520709, 0.0, 0.01063646]
, ReLU()), NeuralVerification.Layer{Id, Float64}([0.30263627 -0.23838618 … -0.09
799442 -0.00773645], [0.04735997], Id())]), Hyperrectangle{Float64, Vector{Float
64}, Vector{Float64}}([10.0, 35.0, 2.5, 10.0, 35.0, 8.5, 10.0, 35.0, 18.5, 10.0
 …  38.5, 10.0, 35.0, 44.5, 10.0, 35.0, 54.5, 10.0, 35.0, 60.5], [0.0, 0.0, 2.5,
 0.0, 0.0, 2.5, 0.0, 0.0, 2.5, 0.0  …  2.5, 0.0, 0.0, 2.5, 0.0, 0.0, 2.5, 0.0, 0
.0, 2.5]), Hyperrectangle{Float64, Vector{Float64}, Vector{Float64}}([-500.0005]
, [499.9995]))

julia> res = solve(ReluVal(max_iter=100), prob)


CounterExampleResult(:violated, [10.0, 35.0, 2.5, 10.0, 35.0, 8.5, 10.0, 35.0, 1
8.5, 10.0  …  38.5, 10.0, 35.0, 44.5, 10.0, 35.0, 54.5, 10.0, 35.0, 60.5])

julia> NeuralVerification.compute_output(net, res.counter_example)
1-element Vector{Float64}:
 4.60099401966847

julia>

julia>
```