# General **Cutting Planes** for Bound-propagation Based **Neural Network Verification**

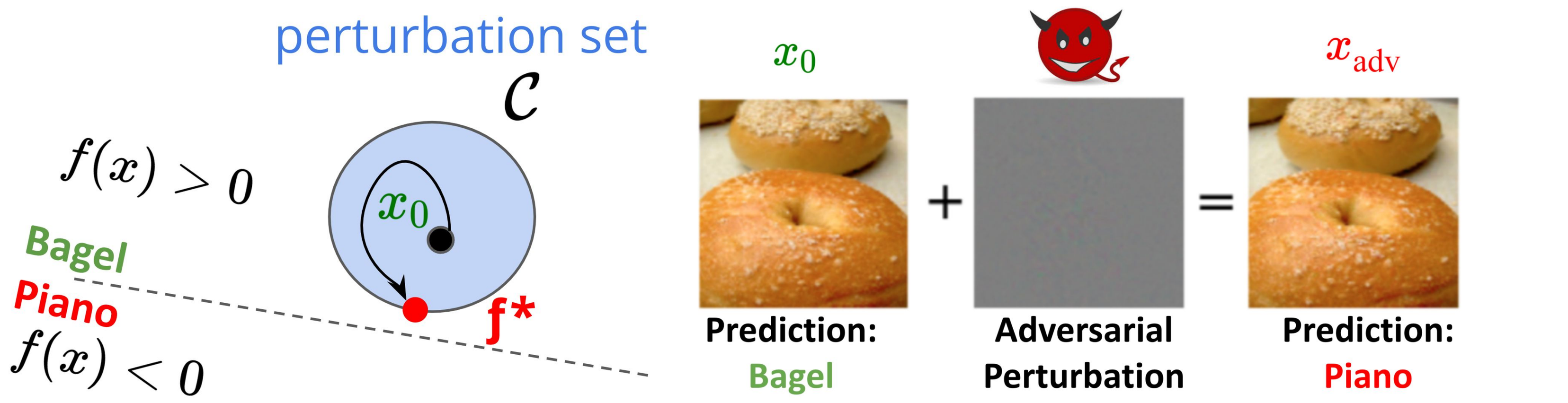
**Huan Zhang*** **Shiqi Wang*** **Kaidi Xu*** Linyi Li Bo Li Suman Jana Cho-Jui Hsieh Zico Kolter

*Equal contribution

CMU Columbia University Drexel University UCLA University of Illinois Urbana-Champaign

Neural Information Processing Systems

## Neural Network Verification (example: robustness)




**Q:** Does the classifier always predicts positive anywhere in the ball?

**Mathematically:** solve $f^* = \min_{x \in \mathcal{C}} f(x)$; positive $f^*$ => verified

**Difficulty:** non-convex due to ReLUs

## A Classical Approach: Mixed Integer Programming [1]

**Weakness:** not optimized for neural network & hard to parallelize

**=> very slow, hardly scale up to large networks**

$f^* = \min x^{(L)}$ Obj: last layer output at layer $L$

s.t. $x^{(i)} = W^{(i)}\hat{x}^{(i-1)} + b^{(i)}$ $i \in \{1, \cdots, L\}$ Linear constraints

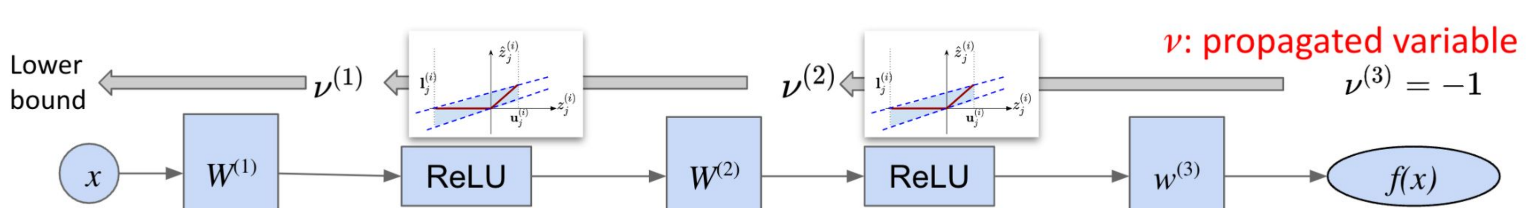
$\hat{x}^{(i)} = \sigma(x^{(i)})$ $i \in \{1, \cdots, L-1\}$ ReLU can be encoded using an integer variable $z^{(i)} \in \{0,1\}$

$\hat{x}^{(0)} = x, \quad x \in \mathcal{C}$ Input set

## SOTA: Bound-Propagation-Based Verifiers

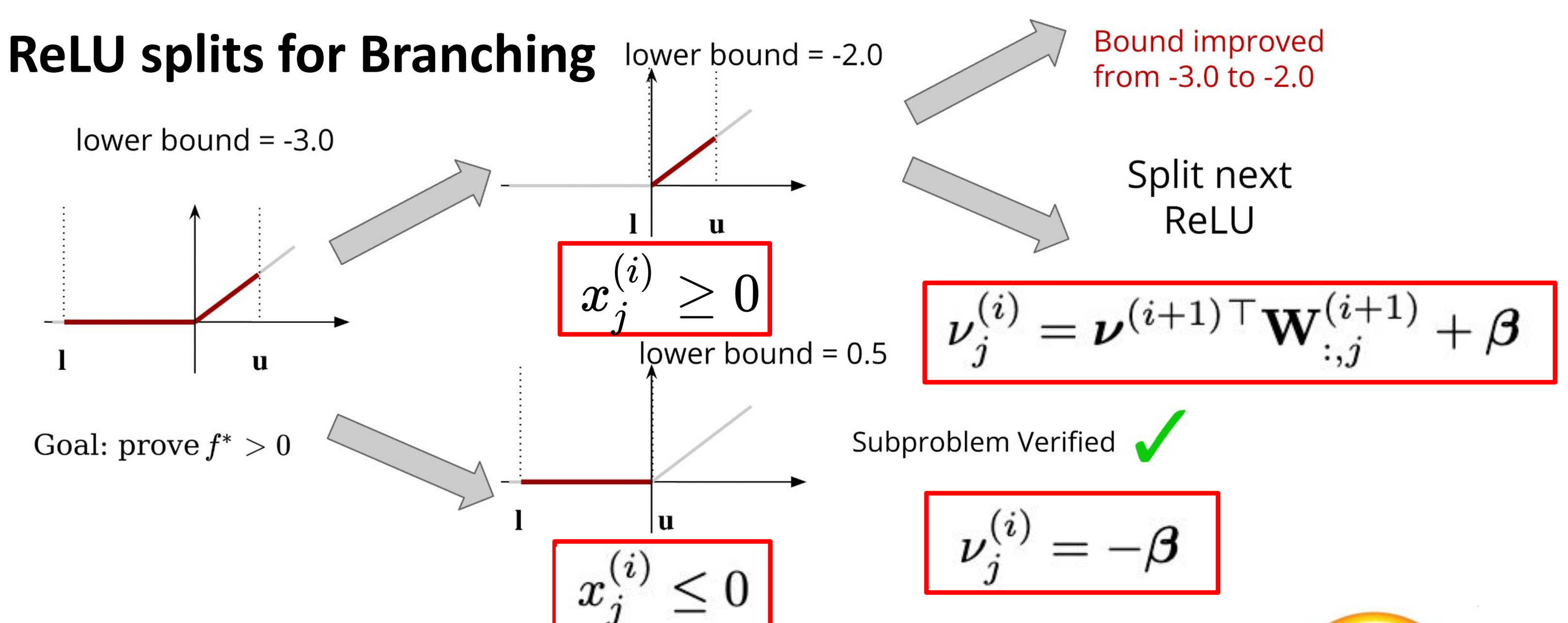
**CROWN [2]:** Propagating **linear bounds** backwards on **GPUs**

- Solves a lower bound of $f^*$ by relaxing ReLU with linear bounds (same tightness as relaxing MIP -> LP)
- Exploiting problem structure; **no LP solver is needed**




**β-CROWN [3]:** bound propagation + branch and bound (BaB)

- Iteratively improves $f^*$ using BaB with additional split constraints
- handles splits via new variable $\beta$ in bound propagation




**Weakness:** fast on most instances but cannot solve some **hard instances** - need **tighter bounds!**

## Why Adding General Cutting Planes?

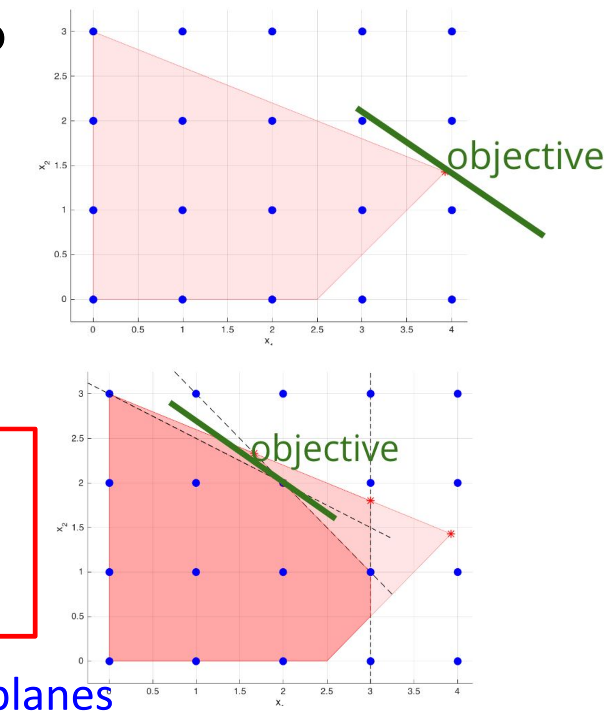
Can produce tighter bounds on linear relaxations of the MIP problem

**Existing bound-propagation based verifier cannot handle cutting plane constraints!**

Additional $N$ cutting plane constraints:

$$\sum_{i=1}^{L-1}\left(\boldsymbol{H}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{G}^{(i)}\hat{\boldsymbol{x}}^{(i)} + \boldsymbol{Q}^{(i)}\boldsymbol{z}^{(i)}\right) \leq \boldsymbol{d}$$

Relaxed integer variables in cutting planes




## GCP-CROWN: Bound-propagation with general cutting plane constraints for tighter bounds

**Goal:** Incorporating cutting planes to bound-propagation methods

**Theorem 3.1** (Bound propagation with general cutting planes). *Given any optimizable parameters* $0 \leq \alpha_j^{(i)} \leq 1$ *and* $\boldsymbol{\beta} \geq 0$, $f^*_{LP\text{-}cut}$ *is lower bounded by the following objective function,* $\pi_j^{(i)*}$ *is a function of* $\boldsymbol{Q}_{:,j}^{(i)}$:

Optimizable variables $\alpha, \beta$

$$g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\epsilon\|\boldsymbol{\nu}^{(1)\top}\mathbf{W}^{(1)}\boldsymbol{x}_0\|_1 - \sum_{i=1}^{L}\boldsymbol{\nu}^{(i)\top}\mathbf{b}^{(i)} - \boldsymbol{\beta}^\top\boldsymbol{d} + \sum_{i=1}^{L-1}\sum_{j \in \mathcal{I}^{(i)}} h_j^{(i)}(\boldsymbol{\beta})$$

*where variables* $\boldsymbol{\nu}^{(i)}$ *are obtained by propagating* $\boldsymbol{\nu}^{(L)} = -1$ *throughout all* $i \in [L-1]$:

$\nu$: propagated variable in bound propagation

$$\nu_j^{(i)} = \boldsymbol{\nu}^{(i+1)\top}\mathbf{W}_{:,j}^{(i+1)} - \boldsymbol{\beta}^\top(\boldsymbol{H}_{:,j}^{(i)} + \boldsymbol{G}_{:,j}^{(i)}), \quad j \in \mathcal{I}^{+(i)}$$

$$\nu_j^{(i)} = -\boldsymbol{\beta}^\top\boldsymbol{H}_{:,j}^{(i)}, \quad j \in \mathcal{I}^{-(i)}$$

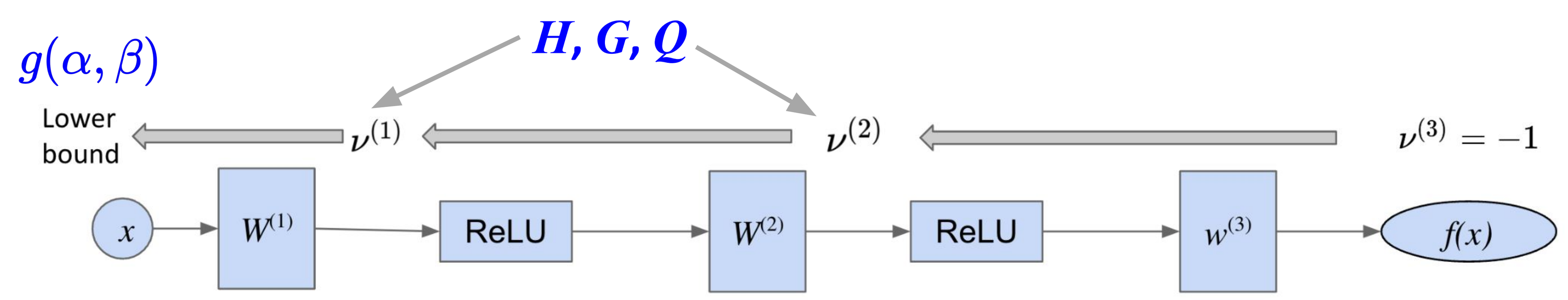
$\boldsymbol{\pi}$ is a function of $\boldsymbol{Q}$

$$\nu_j^{(i)} = \pi_j^{(i)*} - \alpha_j^{(i)}[\hat{\nu}_j^{(i)}]_- - \boldsymbol{\beta}^\top\boldsymbol{H}_{:,j}^{(i)}, \quad j \in \mathcal{I}^{(i)}$$

Cutting plane coefficients (previous works as special cases: H, G, Q being 0)

*Here* $\hat{\nu}_j^{(i)}$, $\pi_j^{(i)*}$ *and* $h_j^{(i)}(\boldsymbol{\beta})$ *are defined for each unstable neuron* $j \in \mathcal{I}^{(i)}$.

### Optimizable cutting plane variable β & cutting plane coefficients $H, G, Q$




**CROWN as a special case:** $H, G, Q$ are 0

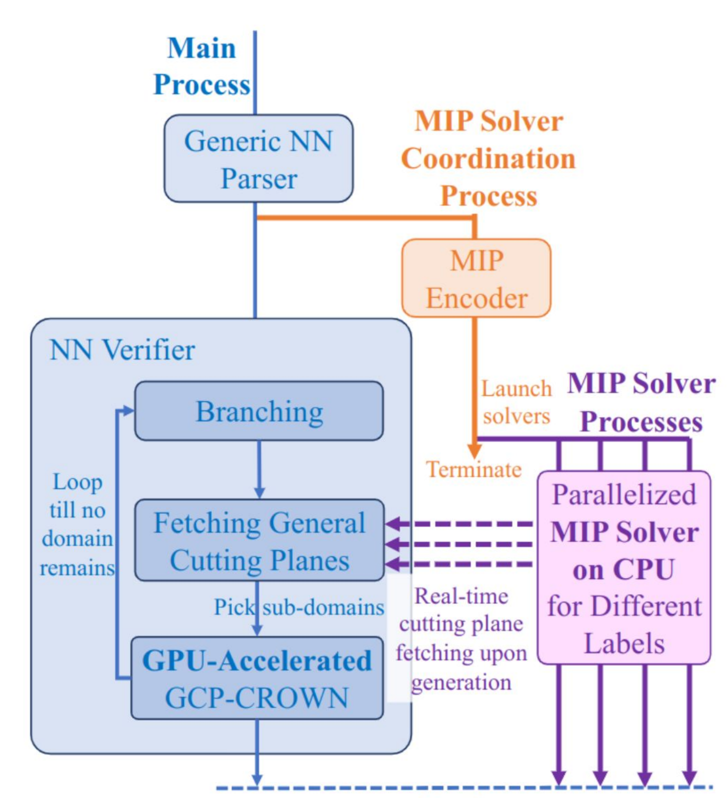
**β-CROWN as a special case:** constraints only with $x_j^{(i)} \geq 0$ or $x_j^{(i)} \leq 0$ (special form of $H$, and $G, Q$ are 0)

## GCP-CROWN Design

**Main thread:** bound propagation on GPU, with cutting planes

**Parallel thread:** A MIP solver on CPU, just for finding cutting planes

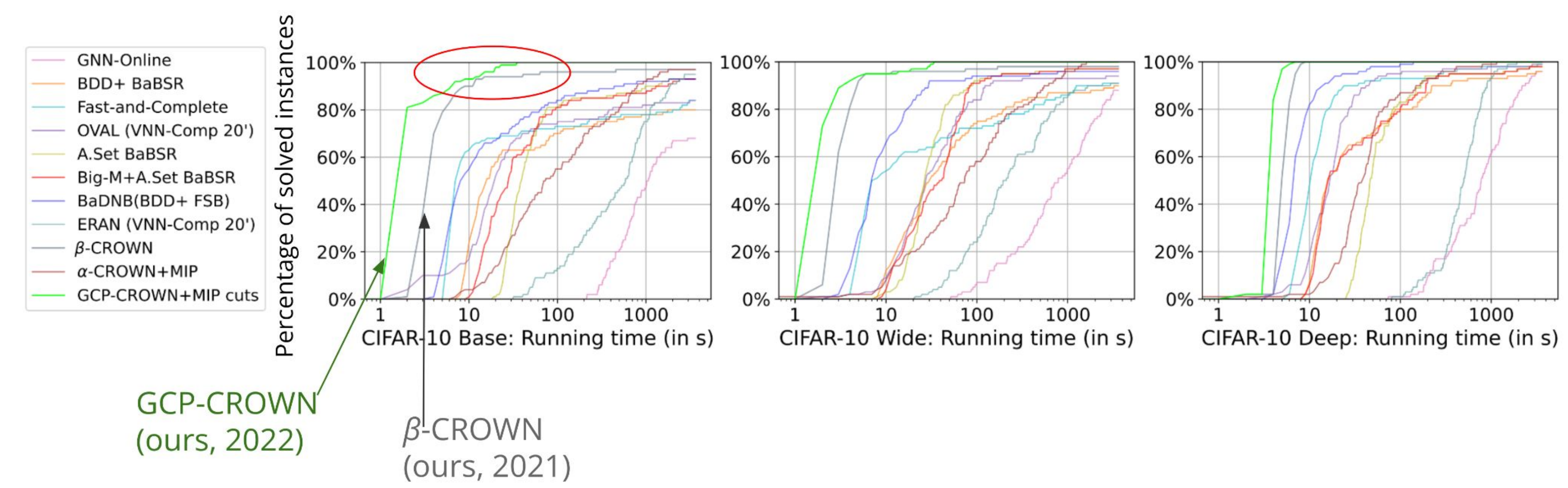
**Future work:** more efficient ways to find cutting planes (valid H, G, Q), specialized for NN verification




## Results: VNN-COMP 2020 (oval20)

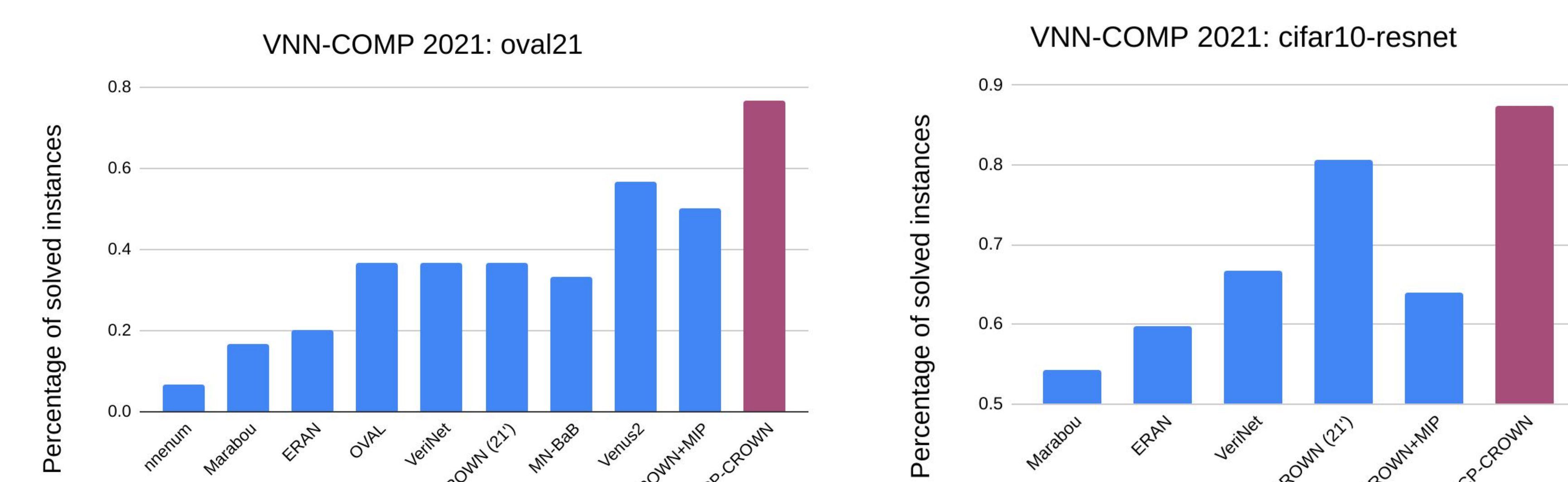
Completely solved (no timeout **for the first time in literature**)

$\beta$-CROWN (VNNCOMP-21 winner) cannot solve 3 hard instances in 1hr




## VNN-COMP 2021 (oval21 & cifar10-resnet)

Almost 2x instances verified than $\beta$-CROWN (VNN-COMP-2021 winner)




## VNN-COMP 2022 (latest)

GCP-CROWN has been incorporated into our tool

**α,β-CROWN: winner of VNN-COMP 2021, 2022**

α,β-CROWN is a versatile NN verifier for verifying robustness and other NN properties

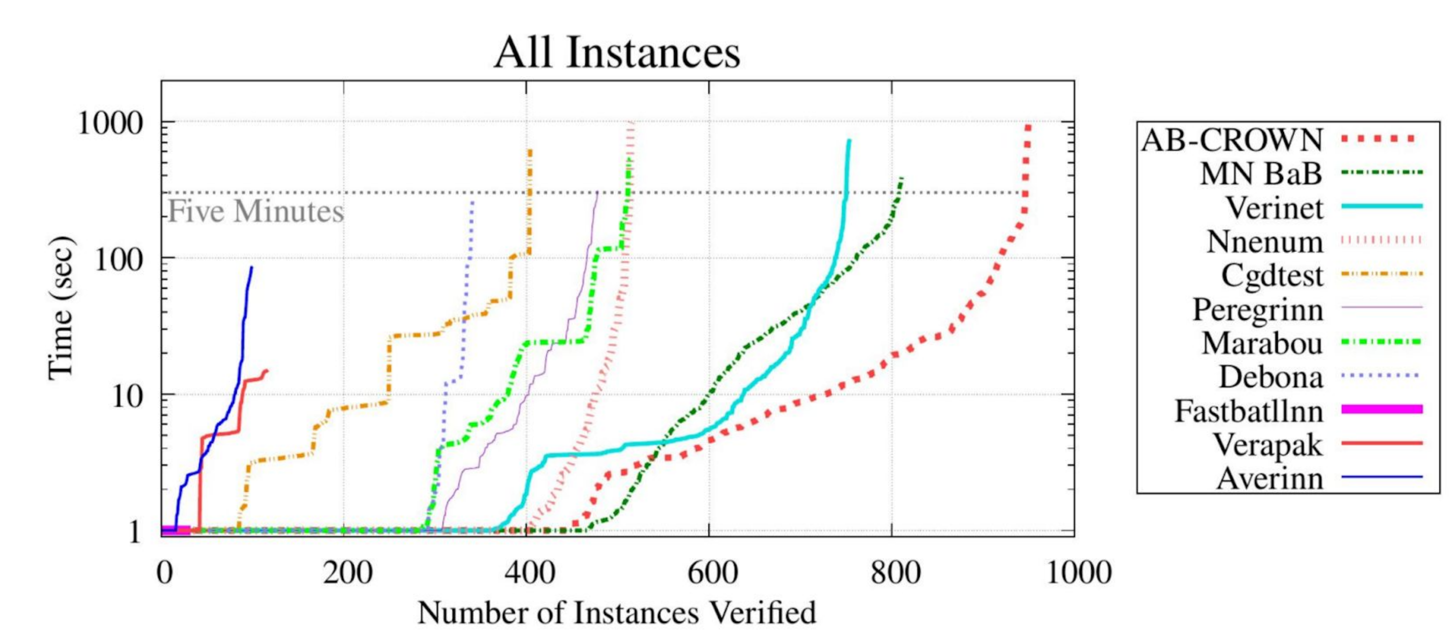
**Try it today: abcrown.org**

αβ CROWN

**Winner of International Verification of Neural Networks Competitions (VNN-COMP 2021,2022)**

Total Score

| # | Tool | Score |
|---|---|---|
| 1 | α,β-CROWN | 1274.9 |
| 2 | MN BaB | 1017.3 |
| 3 | Verinet | 892.5 |
| 4 | Nnenum | 534.0 |
| 5 | Cgdtest | 406.4 |
| 6 | Peregrinn | 399.0 |
| 7 | Marabou | 380.6 |
| 8 | Debona | 222.9 |
| 9 | Fastbatllnn | 100.0 |
| 10 | Verapak | 98.2 |
| 11 | Averinn | 29.1 |




[1] Evaluating Robustness of Neural Networks with Mixed Integer Programming, ICLR'18
[2] Efficient Neural Network Robustness Certification with General Activation Functions, NeurIPS'18
[3] Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Complete and Incomplete Neural Network Verification, NeurIPS'21