Verifying Neural Networks with Reinforcement Learning

Hai Duong[1], Thanh Le, ThanhVu Nguyen[1]

[1]George Mason University

NeurIPS 2026

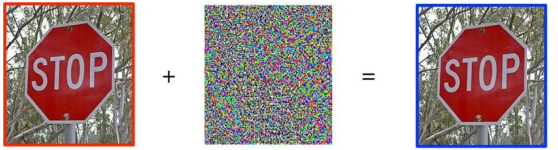




Motivation



DNNs fail in the real world



a tiny perturbation flips the prediction

- DNN verification proves a DNN is safe for *all* perturbations in a continuous interval.
- But verification is expensive: the search space grows exponentially with the model size.

Branch-and-Bound

BaB loop

- **Bounding**: abstraction prunes subproblems.
- **Branching**: pick an unstable ReLU neuron, split into *active* / *inactive*.

Branching order affects solving time

Heuristics (BABSR, FSB) score every neuron and pick the argmax:

- hand-crafted heuristics
- greedy: ignores long-term search cost
- cannot learn from verification data

Can we *learn* long-horizon branching policies?

RSB (RL-Guided Smart Branching)

Key idea

Learn a policy that provides a weight a_{ij} to revise the baseline score ω_{ij}:

$$n^* = \underset{i,j}{\text{argmax}}\ \omega_{ij} \odot a_{ij}$$

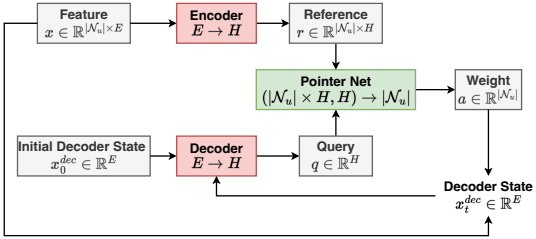
Observation

- List of unstable neurons
- Each unstable neuron: bounds (l, u), bias, mask + GCN embedding.

Reward

-1 per unresolved subproblem $\Rightarrow$ maximizing return = fewer branches.

Architecture and Training



Model

- **PointerNet** actor / twin critics handle a variable number of unstable neurons.
- **Soft Actor-Critic** training.

Contribution

One PointerNet model, trained once, solves various architectures.

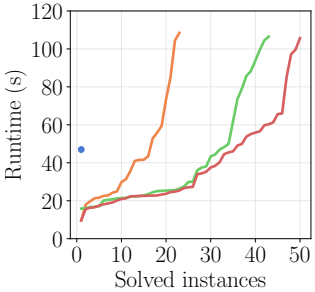
Training

- Trained only on ∼15k hard FNN instances.

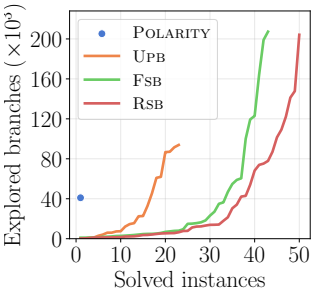
Testing

- Tested on FNN.
- Unseen architectures: CNN, FNN with Sigmoid.

Results on Seen Networks (FNN)



(a) Total runtime

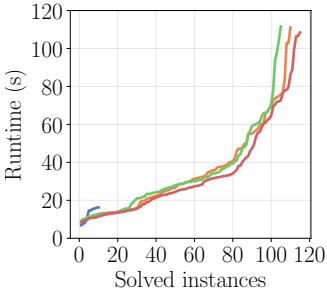


(b) Explored branches

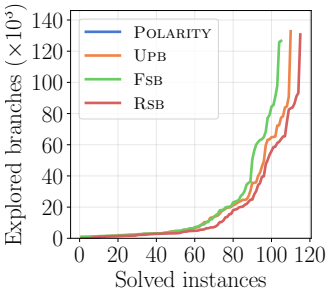
vs. Fsb (SOTA)

FNN: **+16%** instances solved, **−69%** branches, **−22%** time

Results on Unseen Networks (CNN, Sigmoid)



(a) Total runtime



(b) Explored branches

Never trained on these architectures

RSB matches or beats the best baseline, with fewer branches than FSB

Conclusion

- RL branching learns policies that improve upon established heuristics.
- RL captures long-term search strategies, unlike greedy heuristics.
- As RL algorithms advance and training data grows, performance can continue to improve.
- **Future work:** extend RSB to input-space branching and online adaptation during verification.

Reference



1. RSB: `https://github.com/dynaroars/rl-smart-branching`
2. NEURALSAT: `https://github.com/dynaroars/neuralsat`
3. ROARS Lab, GMU: `https://roars.dev`