Destabilizing Neurons to Generate Challenging Neural Network Verification Benchmarks

Linhan Li lli34@gmu.edu
ThanhVu Nguyen tvn@gmu.edu

ROARS Lab @ GMU
https://roars.dev

INTRODUCTION

Neural Network Verification (DNNV) has made significant progress in recent years, with the development of numerous verification techniques and tools. However, the field still lacks high-quality benchmarks for systematically evaluating and improving these tools. As verification techniques advance, many existing benchmarks have become too trivial, while the harder ones often remain unsolvable. Several recent efforts have attempted to address this gap, typically by retraining or distilling neural networks to create new benchmarks. However, such approaches are computationally expensive and often produce benchmarks with unknown or unverifiable ground truth.

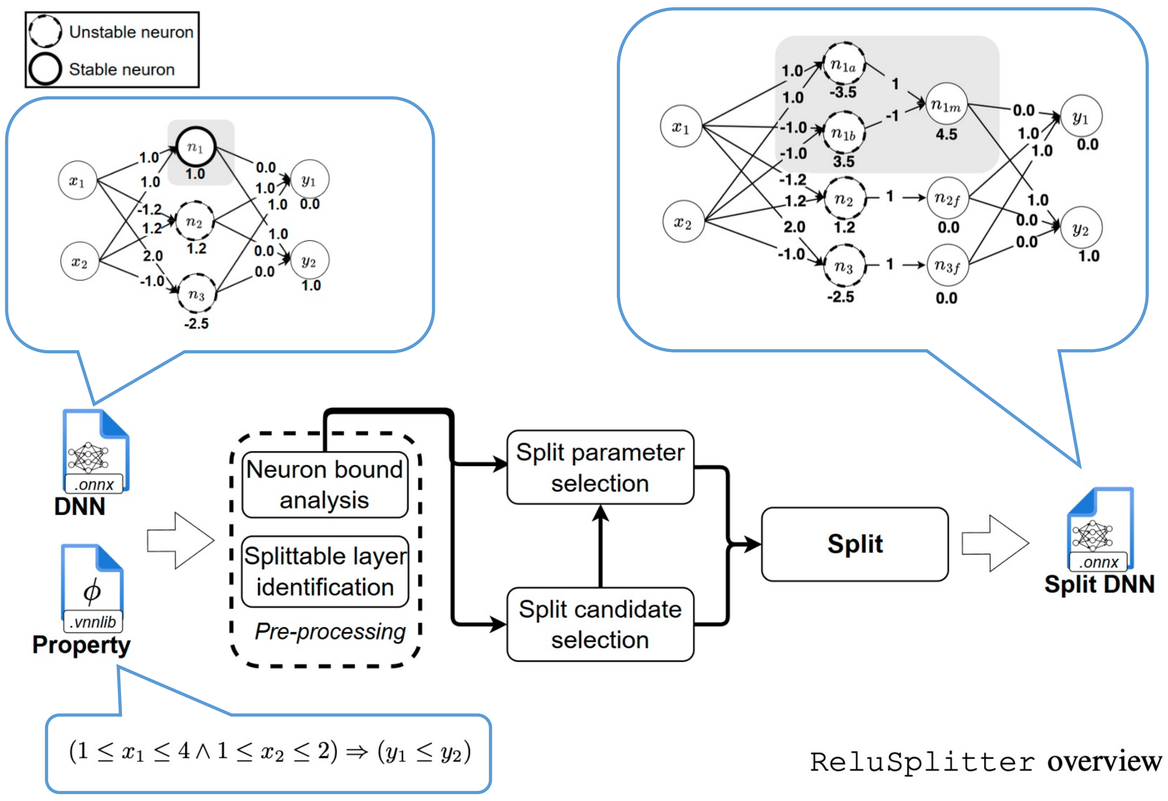
In this paper, we introduce *ReluSplitter*, an automatic benchmark generation tool for DNN verifiers. *ReluSplitter* takes existing verification benchmarks as input and strategically destabilizes stable neurons to increase verification difficulty. This transformation is semantics-preserving by construction: every *ReluSplitter*-generated benchmark is guaranteed to have exactly the same ground truth as the original benchmark. This makes *ReluSplitter* particularly valuable for assessing verifier correctness and performance.

Evaluation shows *ReluSplitter* effectively challenges state-of-the-art DNN verifiers on widely-used benchmarks, consistently increasing the verification time by up to 185 times. It also provided new insights on verifier scalability, implementation robustness, heuristic effectiveness, and optimization opportunities.

Contributions

ReluSplitter makes the following key contributions to the DNNV community:

- It provides a **training-free** method for generating DNNV benchmarks, **eliminating the need for training data.**
- It provides a fast way to reuse well-established DNNV benchmarks to **generate new benchmarks in seconds**.
- It provides **fine-grained difficulty control** over the generated benchmarks by adjusting the number of neurons to destabilize.
- It provides a **semantic-preserving** way to generate new benchmarks that always have the same ground truth as the originals.



`ReluSplitter` overview

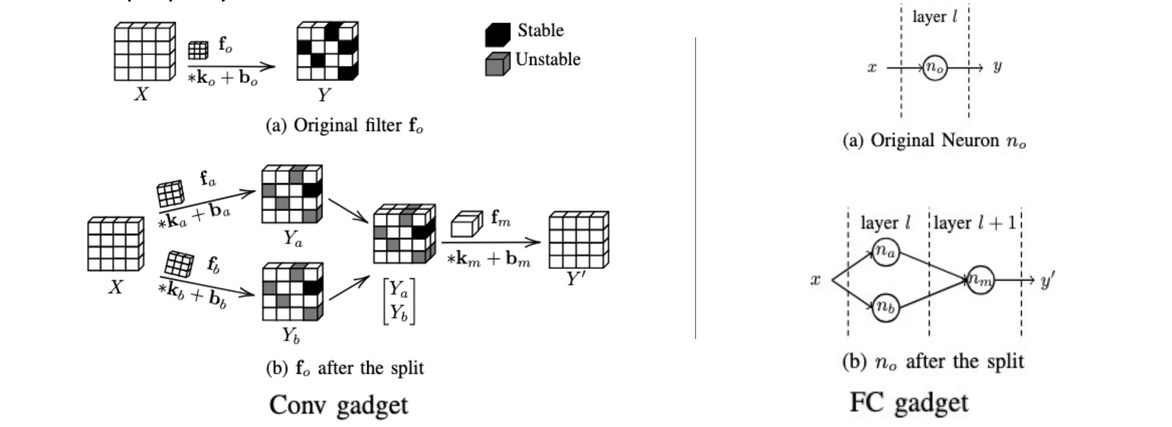
Approach

The complexity of a DNNV problem is closely related to the stability of ReLU neurons.

- **Stable neuron**: Fully linear, simplifies the verification process.
- **Unstable neuron**: Non-linear, makes verification difficult.

ReluSplitter utilize this key idea to generate DNNV benchmarks. It increases the number of unstable neurons in the DNN to create non-linearities, thus making the DNNV problem more challenging.

Given a **DNNV problem** (a **DNN** and **property** pair), *ReluSplitter* uses **linear bound propagation** to identify **stable ReLU neurons** (or **filters**) in the DNN with respect to the property. It then destabilizes them with the **split gadgets.** This transformation results in a **split DNN** that is semantically equivalent to the original DNN, but is much harder to verify against the same property.



In this work, we design two variants of the split gadget: the **FC gadget** and the **Conv gadget**, which are specifically designed for use in fully-connected and convolutional layers. These gadgets can be configured to perfectly match the behavior of any neuron or convolution filter while introducing neuron instability with mathematical guarantee.

Evaluation

Benchmark	Type	# ReLUs	# Instances	Timeout		Verifiers used
				original	generated	
ACAS Xu	FC	300	156	60s	180s	$\alpha\beta$-CROWN, Marabou, NeuralSAT, nnenum
MNIST_FC	FC	0.5-1.5K	41	120s	360s	$\alpha\beta$-CROWN, Marabou, NeuralSAT, nnenum
Oval21	CNN	0.6-2K	86	120s	360s	$\alpha\beta$-CROWN, NeuralSAT
SRI ResNet A/B	ResNet+CNN	11K	67	120s	360s	$\alpha\beta$-CROWN, NeuralSAT

4 SOTA verifiers:

αβ-CROWN, **Marabou**, **NeuralSAT**, **nnenum**.

4 VNN-COMP benchmarks:

ACAS Xu, **MNIST_FC**, **Oval21**, **SRI ResNet A/B**

Easy Instances (verifiable within original timeout) from each benchmark were kept, and harder ones were discarded. For evaluation, each remaining instance was used in three versions:

① Original: The original, unmodified DNNV instance from the benchmarks.

② ReluSplitter: Generated by applying ReluSplitter to ① to **maximize neuron instability**.

③ Baseline: Same network structure as ② but **maintain the neuron stability** of ①.

Verification time on all three versions was recorded. ① uses the original timeout; ② and ③ use a three times longer timeout. Slowdown is then computed for each instance as:

$$\text{Slowdown} = \frac{\text{Verification Time (generated instance)}}{\text{Verification Time (original instance)}} - 1$$

$$\Delta_{SD} = \text{Slowdown}_{\texttt{ReluSplitter}} - \text{Slowdown}_{\text{baseline}}$$

Results

Tab. II: Slowdown (avg./med./max.) for *baseline*, `ReluSplitter`, and *net slowdown* (Δ_{SD}), per benchmark and verifier.

Benchmark	Instance	Verifier	Slowdown (avg./med./max.) Baseline	`ReluSplitter`	Δ_{SD}
ACAS Xu	156	$\alpha\beta$-CROWN	-0.07x / -0.08x / 0.22x	-0.02x / -0.00x / 0.46x	0.06x / 0.02x / 0.51x
		Marabou	0.25x / 0.11x / 9.28x	1.84x / 0.42x / 55.59x	1.59x / 0.32x / 55.40x
		NeuralSAT	0.13x / 0.07x / 1.80x	0.85x / 0.19x / 5.72x	0.72x / 0.09x / 5.62x
		nnenum	0.10x / 0.07x / 2.40x	0.41x / 0.15x / 3.91x	0.31x / 0.04x / 3.74x
MNIST_FC	41	$\alpha\beta$-CROWN	0.07x / 0.07x / 0.13x	11.61x / 0.10x / 148.78x	11.54x / 0.02x / 148.69x
		Marabou	-0.23x / -0.22x / -0.02x	5.59x / 0.72x / 19.04x	5.82x / 0.73x / 19.26x
		NeuralSAT	0.11x / 0.07x / 0.74x	5.39x / 0.29x / 65.16x	5.28x / 0.25x / 65.04x
		nnenum	0.58x / 0.58x / 1.02x	6.49x / 3.73x / 53.66x	5.91x / 3.01x / 52.93x
Oval21	86	$\alpha\beta$-CROWN	0.12x / 0.13x / 0.42x	1.99x / 0.87x / 25.16x	1.86x / 0.79x / 24.87x
		NeuralSAT	0.11x / 0.12x / 0.28x	8.83x / 1.63x / 44.52x	8.72x / 1.49x / 44.45x
SRI ResNet A/B	67	$\alpha\beta$-CROWN	3.05x / 0.01x / 15.83x	6.23x / 0.06x / 185.23x	3.17x / 0.00x / 185.25x
		NeuralSAT	0.07x / 0.03x / 0.94x	3.94x / 0.88x / 78.40x	3.87x / 0.68x / 78.37x

1. ***ReluSplitter* substantially boosted the difficulty of DNNV problems, extending the verification time to over 100 times.**

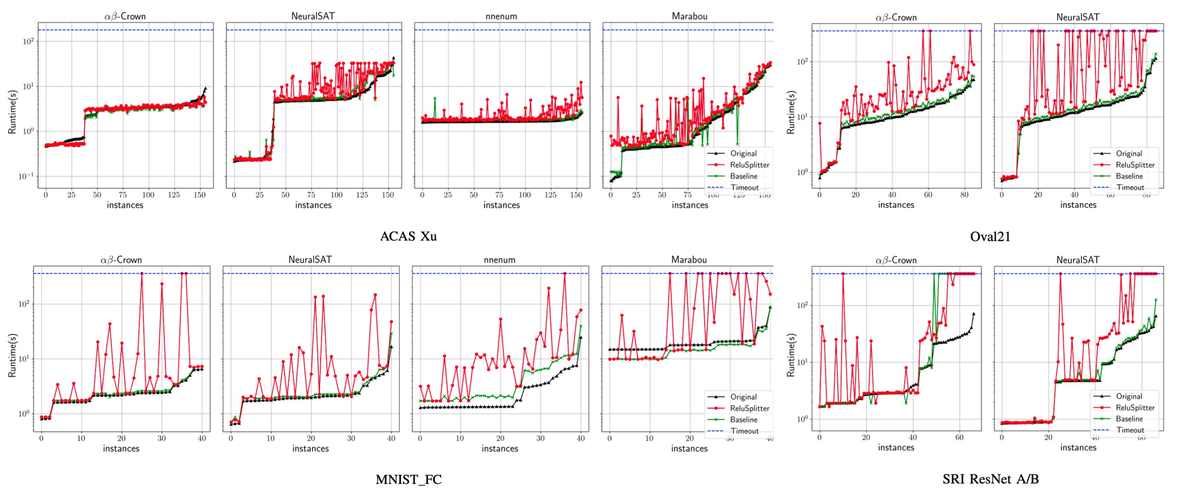


Fig. 5: Runtime comparison. The y-axis (Runtime) is logarithmic; the x-axis represents instances, sorted by the verification time on the original instance.

2. ***ReluSplitter* is an effective testing tool to reveal bugs or unexpected verifier behavior.** It caused *nnenum* to crash on some problems, revealed counterintuitive behavior of *Marabou* on *MNIST_FC* where the larger baseline instances can be verified faster than the originals.

3. ***ReluSplitter* provides insight into verifier evaluation.** It makes scalability difference among verifiers more visible (e.g., on *MNIST_FC*, *Oval21*), and highlight the strengths and weaknesses of different verification approaches (e.g., *αβ-CROWN* and *NeuralSAT,* which share the same abstraction technique but differ in their search heuristic).

Presentation

GitHub Repo

Linhan's Site

ROARS Lab @ GMU