Trustworthy AI (Monsoon 2024)

Assignment 2 - Report

Saumya Mishra

March 16, 2025

Contents

1 Verification of ReLU Networks is NP-Complete

1.1 Problem Definition

We consider the existential variant of ReLU network verification: Given a ReLU network

$$N : \mathbb{R}^n \to \mathbb{R}$$

and an input domain $\mathcal{X} \subseteq \mathbb{R}^n$, decide whether there exists an input $x \in \mathcal{X}$ such that $N(x) \in P$. We reduce 3SAT to this problem.

1.2 Membership in NP

A given input x can be verified in polynomial time by performing a forward pass through N, which consists of polynomially many ReLU operations. Thus, the problem is in NP.

1.3 NP-Hardness via Reduction from 3SAT

1.3.1 Constructing the ReLU Network

Given a 3SAT formula ϕ with variables $x_1, x_2, \ldots, x_n$ and clauses $C_1, C_2, \ldots, C_m$, we construct a ReLU network N_ϕ with:

- n input neurons corresponding to $x_1, \ldots, x_n$.
- Clause evaluation neurons z_j that indicate whether each clause is satisfied.
- A final neuron that aggregates clause satisfaction.
- Additional constraints to enforce binary inputs.

Clause Gadgets: Each clause C_j consists of three literals $\ell_{j1}, \ell_{j2}, \ell_{j3}$. Define:

- If $\ell_{jk} = x_i$, use x_i.
- If $\ell_{jk} = \neg x_i$, use $1 - x_i$.

The clause output is defined as:

$$z_j = \text{ReLU}(\ell_{j1} + \ell_{j2} + \ell_{j3} - 1).$$

If any literal is 1, then $z_j > 0$, otherwise $z_j = 0$.
Final Aggregation: To ensure all clauses are satisfied, define the final output neuron:

$$y = \text{ReLU}\Big(\sum_{j=1}^{m} z_j - (m-1) - \sum_{i=1}^{n} p_i\Big),$$

where p_i penalizes non-binary inputs.

1.3.2 Enforcing Binary Inputs

To ensure $x_i \in \{0, 1\}$, define the penalty term:

$$p_i = \text{ReLU}(x_i - 1) + \text{ReLU}(-x_i).$$

This ensures that $p_i = 0$ only for $x_i = 0$ or $x_i = 1$, and is positive otherwise. The sum $\sum p_i$ in the final neuron ensures that any non-binary assignment leads to $y = 0$.

1.3.3 Correctness of the Reduction

By construction:

- If ϕ is satisfiable, there exists a binary input x such that $y = 1$.
- If ϕ is unsatisfiable, then for every x, at least one $z_j = 0$ or some $p_i > 0$, ensuring $y = 0$.
- The network size is polynomial in ϕ, ensuring a polynomial-time reduction.

1.4 Conclusion

Since ReLU verification is in NP and NP-hard, we conclude that it is NP-complete.

2 Verification Properties

The dataset I selected was a Diabetes prediction dataset, which was selected mainly owing to the simplicity of the input paramters: all the input datapoints are numeric. I trained nine feedforward ReLU networks with varying depth and width. The models follow three structural patterns: (1) **single hidden layer** with 32, 64, or 128 neurons, (2) **two hidden layers** with (32, 32), (64, 64), or (128, 128) neurons, and (3) **three hidden layers** with (32, 32, 32), (64, 64, 64), or (128, 128, 128) neurons. Each model processes structured numerical inputs (e.g., glucose, BMI, age) through ReLU-activated layers and produces a classification probability via a sigmoid-activated output neuron. The models are trained using binary cross-entropy loss and the Adam optimizer for 50 epochs with early stopping. These variations allow us to investigate how different network architectures impact robustness and verifiability using ReLUplex.

2.1 Feature Indices for Reference

Before verifying any properties, the feature indices (excluding the `Outcome` feature) are printed to ensure that the correct indices are used when setting input constraints.

2.2 Property 1: High Glucose Leads to Diabetes Classification

Precondition: The glucose feature (at index 3) is constrained to lie within the high normalized range:

$$x_3 \in [1.5, 3.0].$$

Postcondition: The model is expected to predict diabetes (class 1), meaning:

$$f(x) \geq 0.5.$$

Explanation: This property tests whether an elevated glucose level causes the model to classify the input as diabetic.

2.3 Property 2: Low HDL Leads to Diabetes Classification

Precondition: The HDL feature (at index 7) is set to a low normalized range:

$$x_7 \in [-3.0, -1.5].$$

Postcondition: Under this condition, the model should still predict diabetes (class 1):

$$f(x) \geq 0.5.$$

Explanation: This test verifies that a low HDL level, which is a known risk factor, triggers a positive diabetes classification.

2.4 Property 3: Robustness Around a Non-Diabetic Example

Precondition: For a non-diabetic example x^0 (with class 0), the first five features are perturbed within:

$$x_i \in \left[x_i^0 - 0.1,\ x_i^0 + 0.1\right], \quad \forall i \in \{0, 1, 2, 3, 4\}.$$

Postcondition: The model should maintain its prediction of non-diabetes:

$$f(x) < 0.5.$$

Explanation: This property examines the robustness of the model by checking that small perturbations around a non-diabetic example do not change its classification.

2.5 Property 4: Moderate Values Should Not Give Extreme Predictions

Precondition: Several features are constrained to moderate normalized values:

$$x_3 \in [-0.5, 0.5],$$
$$x_5 \in [-0.5, 0.5],$$
$$x_2 \in [-0.5, 0.5],$$
$$x_8 \in [-0.5, 0.5].$$

Postcondition: The model should predict non-diabetes (class 0):

$$f(x) < 0.5.$$

Explanation: This condition ensures that when the features are within moderate ranges, the model does not output an extreme prediction (i.e., it should not misclassify a moderate case as diabetic).

2.6 Property 5: Multiple Diabetes Indicators Lead to Positive Classification

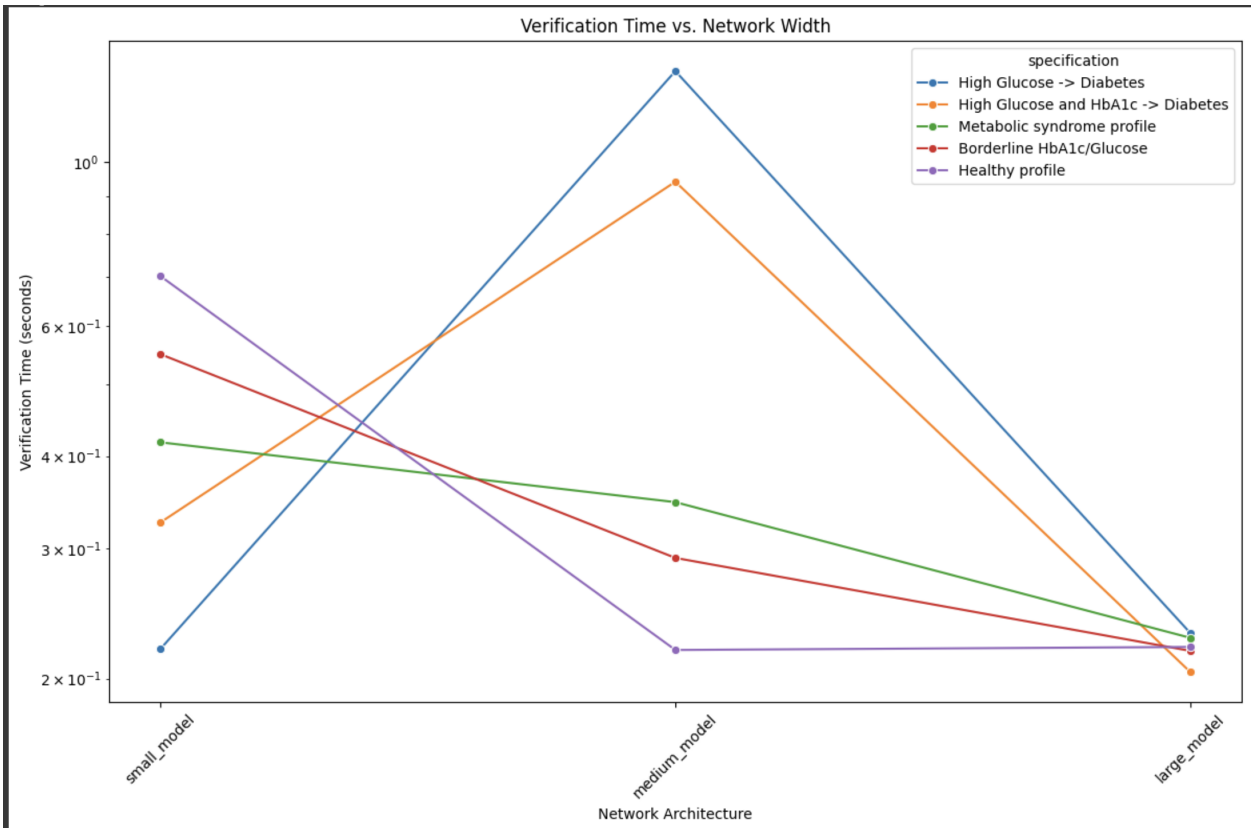
Precondition: Multiple features are set to high normalized values:

$$x_3 \in [1.5, 3.0],$$
$$x_5 \in [1.5, 3.0],$$
$$x_2 \in [1.5, 3.0],$$
$$x_9 \in [1.5, 3.0].$$

Postcondition: The expected outcome is a diabetes prediction (class 1):

$$f(x) \geq 0.5.$$

Explanation: This property checks that the presence of multiple high-risk indicators (as reflected by the specified features) correctly triggers a positive diabetes classification.
The verification results indicate that counterexamples were found for all properties, suggesting that the model's decision boundaries do not align well with the expected behavior under these input constraints. This inconsistency implies that either the model needs further refinement or the property specifications may need to be re-evaluated for better alignment with the actual learned representations.



3 Question 3

The results indicate that regardless of model size (small, medium, or large), Reluplex consistently finds counterexamples for the specified properties. This suggests that either the learned decision boundaries of the models do not align with the desired safe classifications or that the property constraints may be too strict, and that simply increasing depth or width does not mitigate these issues.