Verification of RNN-Based Neural Agent-Environment Systems

Michael Akintunde, Andreea Kevorchian, Alessio Lomuscio, Edoardo Pirovano

Imperial College London, UK

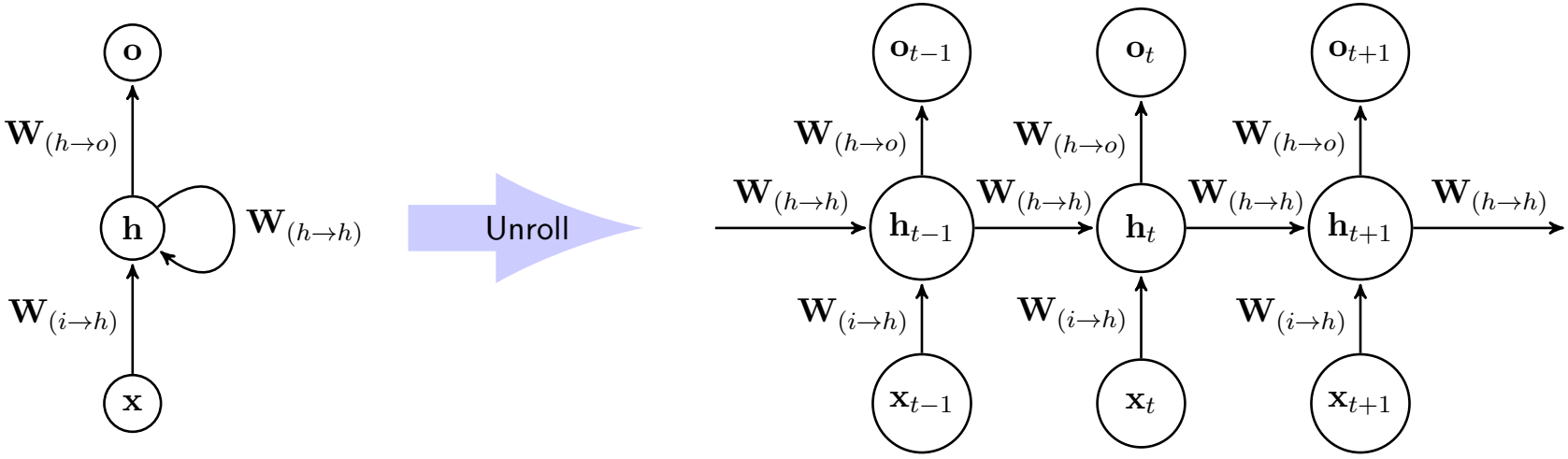
VNN 2019, Stanford, California

This work

- We introduce Recurrent Neural Agent-Environment Systems to formalise **RNN-based agents** interacting with an environment with non-linear dynamics.
- We define and study various verification problems for these systems.
- We define two methods to solve said verification problems.
- We present an implementation and report experimental results.
- The paper builds upon work from previous work (KR'18)

Recurrent Neural Networks (RNNs)

- Many approaches already exist to perform verification on single FFNNs and closed-loop systems with FFNN-based agents.
- RNNs, equipped with a state that evolves over time, are designed to process sequences of data



Single-Layer Recurrent Neural Networks (RNNs)

Definition

A single-layer *recurrent neural network* (RNN) R with h *hidden units* and input size i and output size o is a neural network associated with the weight matrices $\mathbf{W}_{(i\to h)} \in \mathbb{R}^{i\times h}$, $\mathbf{W}_{(h\to h)} \in \mathbb{R}^{h\times h}$ and $\mathbf{W}_{(h\to o)} \in \mathbb{R}^{h\times o}$, and the two activation functions $\sigma : \mathbb{R}^h \to \mathbb{R}^h$ and $\sigma' : \mathbb{R}^o \to \mathbb{R}^o$.

Here we assume the activation functions $\sigma = \sigma' =$ ReLU.

Function Computed by an RNN

Definition (Function computed by RNN)

For an RNN R with weight matrices $\mathbf{W}_{(i\to h)}$, $\mathbf{W}_{(h\to h)}$ and $\mathbf{W}_{(h\to o)}$, let $\mathbf{x} \in (\mathbb{R}^k)^n$ denote an input sequence of length n where each element of the sequence is a vector of size k, with $\mathbf{x}_t$ denoting the t-th vector of $\mathbf{x}$. We define $\mathbf{h}_0^{\mathbf{x}} = \mathbf{0}$ as a vector of 0s. For each time step $1 \leq t \leq n$, we define:

$$\mathbf{h}_t^{\mathbf{x}} = \sigma(\mathbf{W}_{(h\to h)}\mathbf{h}_{t-1}^{\mathbf{x}} + \mathbf{W}_{(i\to h)}\mathbf{x}_t).$$

Then, the *output* of the RNN is given by $f(\mathbf{x}) = \sigma'(\mathbf{W}_{(h\to o)}\mathbf{h}_n^{\mathbf{x}})$.

Recurrent Neural Agent-Environment Systems

Definition (RNN-AES)

A **Recurrent Neural Agent-Environment System** (RNN-AES) is a tuple $AES = (Ag, E, I)$ where:

- Ag is a **recurrent neural agent** with action function $act : O^* \to Act$,
- $E = (S, O, o, t_E)$ is an **environment** with
  - state space $S \subseteq \mathbb{R}^m$,
  - observation space $O \subseteq \mathbb{R}^{m'}$,
  - observation function $o : S \to O$ and
  - transition function $t_E : S \times Act \to S$,
- $I \subseteq S$ is a set of **initial** states.

Paths are sequences of env state observations determined by the transition function t_E from an initial state.

We assume linearly definable AES (both t_E and I).

Bounded Specifications

Definition (Specifications)

For an environment with state space $S \subseteq \mathbb{R}^m$, we consider a fragment of LTL given by the following BNF:

$$\phi ::= X^k \mathcal{C} \mid \mathcal{C} U^{\leq k} \mathcal{C}$$
$$\mathcal{C} ::= \mathcal{C} \vee \mathcal{C} \mid (i)\, op\, (j) \mid (i)\, op\, x$$

where $op \in \{<, \leq, =, \neq, \geq, >\}, i, j \in \{1, \ldots, m\}, x \in \mathbb{R}$, $k \in \mathbb{N}$.

Satisfaction

Satisfaction relation $\models$ is defined as follows:

Definition (Satisfaction)

Given a path $\rho \in \Pi$ on an RNN-AES and a formula ϕ:

$$
\begin{array}{lll}
\rho \models (i)\, op\, (j) & \text{iff} & \rho(0).i \, op \, \rho(0).j \text{ holds;} \\
\rho \models \mathcal{C}_1 \vee \mathcal{C}_2 & \text{iff} & \rho \models \mathcal{C}_1 \text{ or } \rho \models \mathcal{C}_2; \\
\rho \models X^k \mathcal{C} & \text{iff} & \rho(k) \models \mathcal{C}; \\
\rho \models \mathcal{C}_1 U^{\leq k} \mathcal{C}_2 & \text{iff} & \text{there is some } i \leq k \text{ such that } \rho(i) \models \mathcal{C}_2 \text{ and} \\
 & & \rho(j) \models \mathcal{C}_1 \text{ for all } 0 \leq j < i.
\end{array}
$$

Verification problem

We say that an agent-environment system AES satisfies a specification ϕ if it is the case that every path originating from an initial state $i \in I$ satisfies ϕ, denoted $AES \models \phi$.

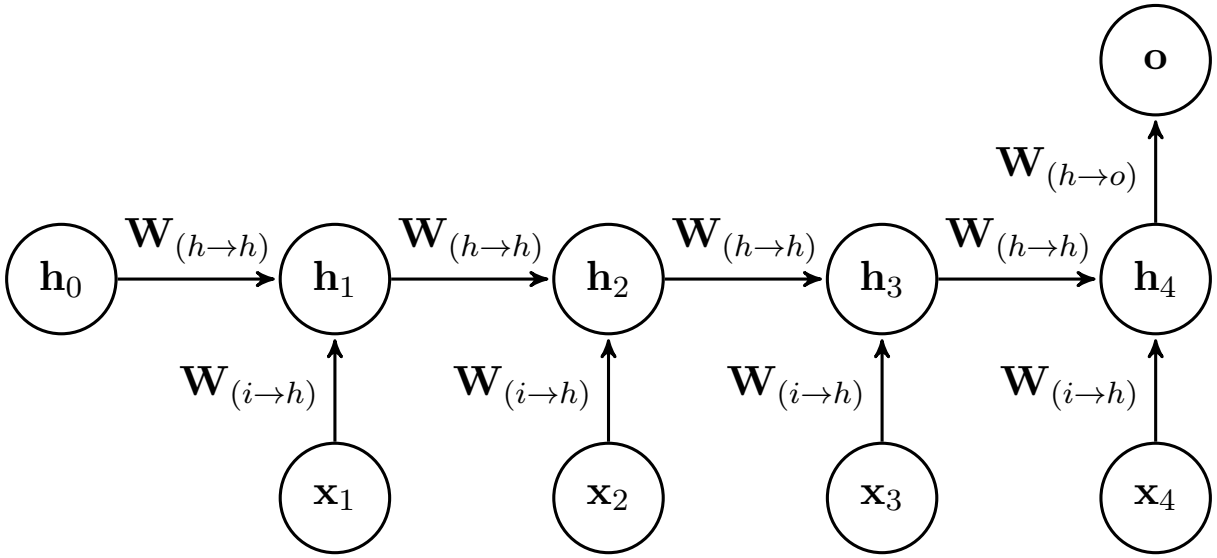
This is the basis of the verification problem:

Definition (Verification problem)

Determine if given an RNN-AES AES and a formula ϕ, it is the case that $AES \models \phi$.

Approach: Unrolling RNNs to FFNNs

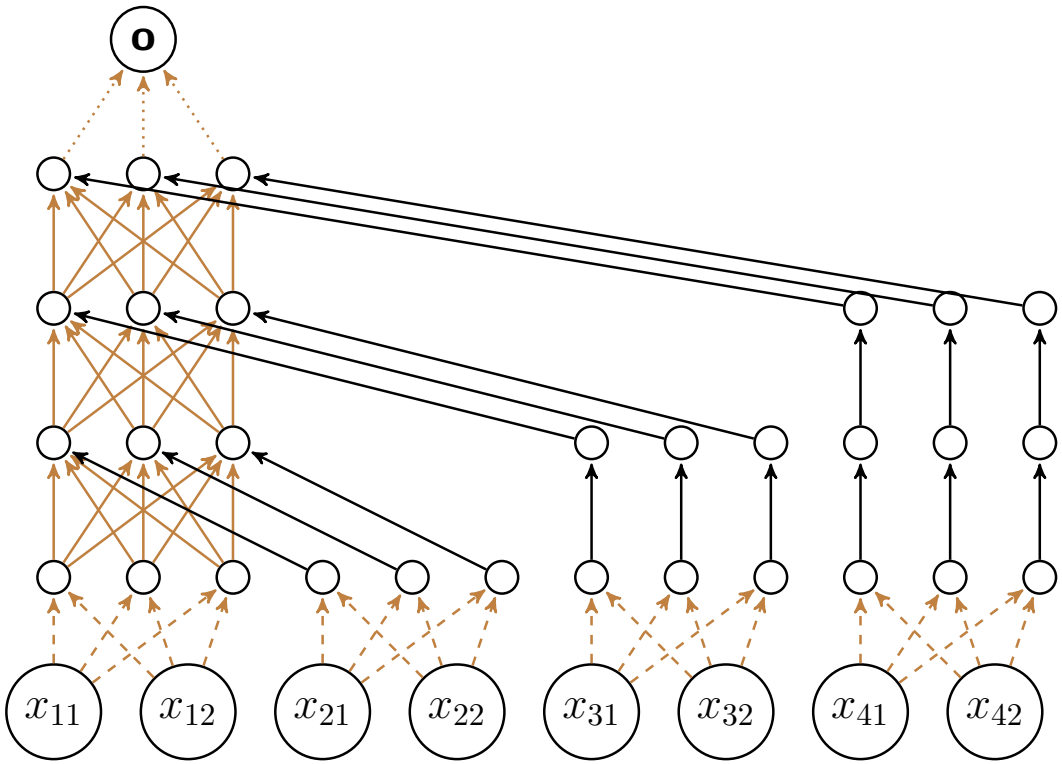
Example: How to construct an FFNN from an RNN with input sequence of length 4, input size of 2, 3 hidden units and output size 1 (single output)?



Approach: Unrolling RNNs to FFNNs

Input on Start (IOS)

Scale input values according to the weights of $W_{(i \to h)}$. At each time step when the input is needed, pass it unchanged to the corresponding hidden layer of the FFNN.

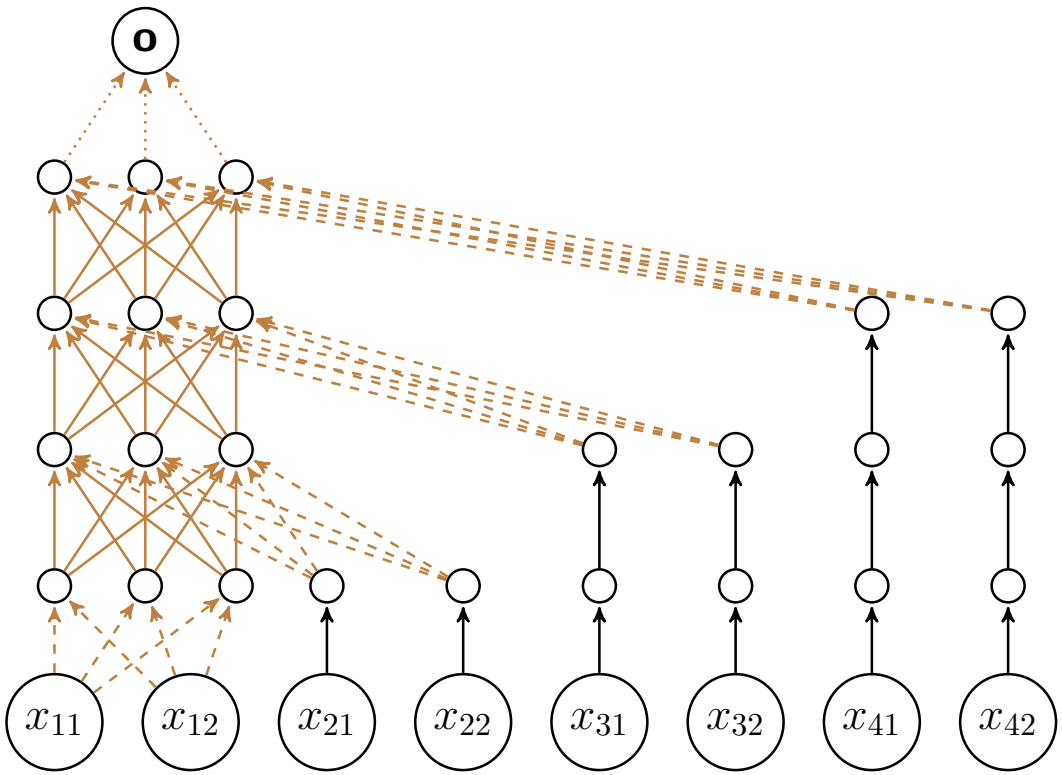


FFNN constructed from RNN with length 4 input sequence, input size of 2, 3 hidden units and output size 1.

Approach: Unrolling RNNs to FFNNs

Input on Demand (IOD)

At the time step when the input term is needed, scale the input (on demand) and pass to the corresponding hidden layer of the FFNN, otherwise propogate the term's original value.



FFNN constructed from RNN with length 4 input sequence, input size of 2, 3 hidden units and output size 1.

Equivalences

Theorem

For an RNN-AES AES and a specification ϕ^k,

$$AES \models \phi^k \text{ iff } IOD(AES) \models \phi^k \text{ iff } IOS(AES) \models \phi^k.$$

Verification on bounded specifications of RNN-AES can be recast as FFNN-AES verification. See paper for further details of the unrolling methods.

Verification for FFNN-AES addressed in KR'18 paper.

MILP Encoding for ReLU-FFNN

Maganti & Lomuscio, 2017, Cheng, Nührenberg & Ruess, 2017

ReLU activation function

$$\mathbf{x}_j^{(i)} = \max\left(0, \mathbf{W}_j^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}_j^{(i)}\right), \quad j = 1 \cdots |L^{(i)}|$$

- Active phase: $\mathbf{x}_j^{(i)} = \mathbf{W}_j^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}_j^{(i)}$ (set $\bar{\delta}_j^{(i)} = 0$)
- Inactive phase: $\mathbf{x}_j^{(i)} = 0$ (set $\bar{\delta}_j^{(i)} = 1$)
- Value of $\bar{\delta}_j$ forces two of the four constraints to become vacuously true, and the other two correspond exactly to inactive/active phase of neuron:

$$\begin{aligned}
\mathbf{x}_j^{(i)} &\geq \mathbf{W}_j^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}_j^{(i)} \\
\mathbf{x}_j^{(i)} &\leq \mathbf{W}_j^{(i)}\mathbf{x}^{(i-1)} + \mathbf{b}_j^{(i)} + M\bar{\delta}_j^{(i)} \\
\mathbf{x}_j^{(i)} &\geq 0 \\
\mathbf{x}_j^{(i)} &\leq M(1 - \bar{\delta}_j^{(i)})
\end{aligned}$$

Verifying RNN-AESs via MILP

Theorem

The MILP P_{FFNN} is feasible for $\bar{x}^{(1)} = \bar{x}, \bar{x}^{(m)} = \bar{y}$ iff $f_{NN}(\bar{x}) = \bar{y}$.

Verification problem can be solved via MILP by considering the linear programming problem defined on the unrolled RNN truncated by the bound on the spec.

Theorem

Verification of RNN-AESs against bounded specifications is coNP-complete.

Verification Procedure

Goal: Take RNN-AES $AES = (Ag_N, \overbrace{(S, O, o, t_E)}^{\text{Environment } E}, I)$ and a specification ϕ. Return whether ϕ is satisfied on the system.

For $X^k\mathcal{C}$:

- For each step n from $0 \to k$, add constraints corresponding to the observation function, the unrolling of length n of the RNN and the transition function of the environment
- Check whether $\bar{\mathcal{C}}$ can be satisfied in any of the states possible after k steps, and return result accordingly.

Verification Procedure

For $\mathcal{C}_1 U^{\leq k} \mathcal{C}_2$:

- For each n from 0 to k, check whether $\mathcal{C}_2$ is always satisfied in valid paths of length n that have not already had C_2 satisfied earlier on. If so, return `True`.
- Otherwise, continue from states not satisfying $\mathcal{C}_2$. Check if not all of these satisfy $\mathcal{C}_1$. If so, return `False`.
- Otherwise, we're on a valid path. Continue to add the constraints corresponding to the observation function, the unrolling of length n of the RNN and the transition function of the environment. Iterate to $n + 1$.
- If reached $n = k$ without a result returned, there must exist a path of length k along which C_2 is never satisfied, and so we return `False`.

RNSVerify

- Experimental toolkit produced, solving desired verification problems.
- Takes as input an RNN-AES, property ϕ and produces associated MILP problem.
- Fed to Gurobi 7.5.2 to solve.
- If output is `False`, counterexample in the form of a trace is shown.

Example: OpenAI Pendulum

Brockman et. al, 2016

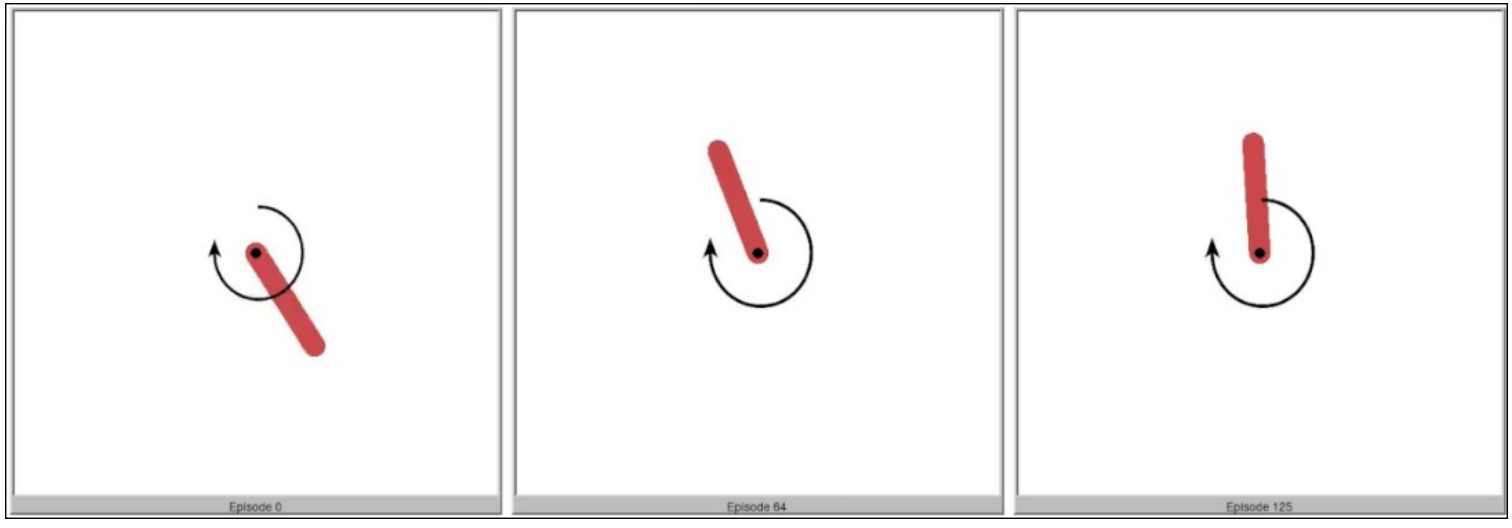
Example (Pendulum)

OpenAI Gym task PENDULUM-V0:

- System composed of a pendulum and an agent which can apply a force to the pendulum.
- Agent can observe the current angle θ of the pendulum ($\theta = 0$ indicates that it is perfectly vertical) and the pendulum's angular velocity $\dot{\theta}$.
- Agent chooses a small **torque** to be applied to the pendulum at each time step.
- **Aim**: Learn how to keep the pendulum upright by applying torque at each time step.

Example: OpenAI Pendulum

Brockman et. al, 2016



Evaluation: OpenAI Pendulum

Agent observes the angle and angular velocity and applies a torque to keep it vertical.

- Encoded as a RNN-AES: agent-environment system, non-linear transition function, and sequence of env state observations.
- Agent's policy synthesised using Q-Learning on a ReLU-RNN. Env approximated from data (since env is non linear).

RNSVerify found several bugs in the synthesised agent, e.g., the agent would apply the torque incorrectly in some situations.

Verification Results

Input on Start – Evaluation on Pendulum [OpenAI, 2018]

Check the property $X^n(\theta_f > -\varepsilon)$ for different values of n and ε using IOS. Fix $(\theta_i, \dot{\theta}_i) \in [0, \pi/64] \times [0, 0.3]$.

		ε			
		$\pi/10$	$\pi/30$	$\pi/50$	$\pi/70$
n	1	0.056s	0.067s	0.011s	0.014s
	2	0.052s	0.179s	0.138s	0.197s
	3	0.372s	0.904s	5.794s	0.552s
	4	2.578s	7.222s	0.378s	0.368s
	5	20.57s	31.07s	0.748s	0.663s
	6	73.97s	3.264s	31.07s	23.99s
	7	54.30s	96.54s	116.8s	207.8s
	8	693.2s	294.9s	239.8s	243.3s

Greyed areas denote `False` result, hence insufficiently trained system.

Verification Results

Input on Demand – Evaluation on Pendulum [OpenAI, 2018]

Check the property $X^n(\theta_f > -\varepsilon)$ for different values of n and ε using IOD. Fix $(\theta_i, \dot{\theta}_i) \in [0, \pi/64] \times [0, 0.3]$.

		ε			
		$\pi/10$	$\pi/30$	$\pi/50$	$\pi/70$
n	1	0.004s	0.012s	0.011s	0.014s
	2	0.060s	0.114s	0.244s	0.253s
	3	0.247s	1.068s	6.092s	0.125s
	4	2.176s	5.359s	0.182s	0.198s
	5	10.04s	0.293s	0.317s	0.294s
	6	13.99s	0.367s	0.357s	0.359s
	7	31.93s	0.497s	0.488s	0.478s
	8	0.689s	0.660s	0.696s	0.703s

Greyed areas denote `False` result, hence insufficiently trained system.

Number of Constraints and Variables

n	Input on Start		Input on Demand	
	V	C	V	C
1	273	336	273	336
2	736	736	620	766
3	1455	1806	1055	1306
4	2494	3101	1590	1971
5	3917	4876	2237	2776
6	5788	7211	3008	3736
7	8171	10186	3915	4866
8	11130	13881	4970	6181

Table: For different values of n, size of constraint problem constructed by $\mathrm{RNSVerify}$ w.r.t number of variables (V) and constraints (C) when checking $X^n(\theta_f > -\varepsilon)$. We observe a degradation in performance with the length of the paths.

Conclusions

- Increased attention to verifiable AI.
- First approach on verification of a closed-loop system composed of a neural agent based on an ReLU-RNN.
- Sound and complete procedure produced, effective for controllers of limited complexity.
- Approach is independent of the underlying solver.