Reachability Analysis for Neural Agent-Environment Systems

M. Akintunde, A. Lomuscio, L. Maganti, E. Pirovano

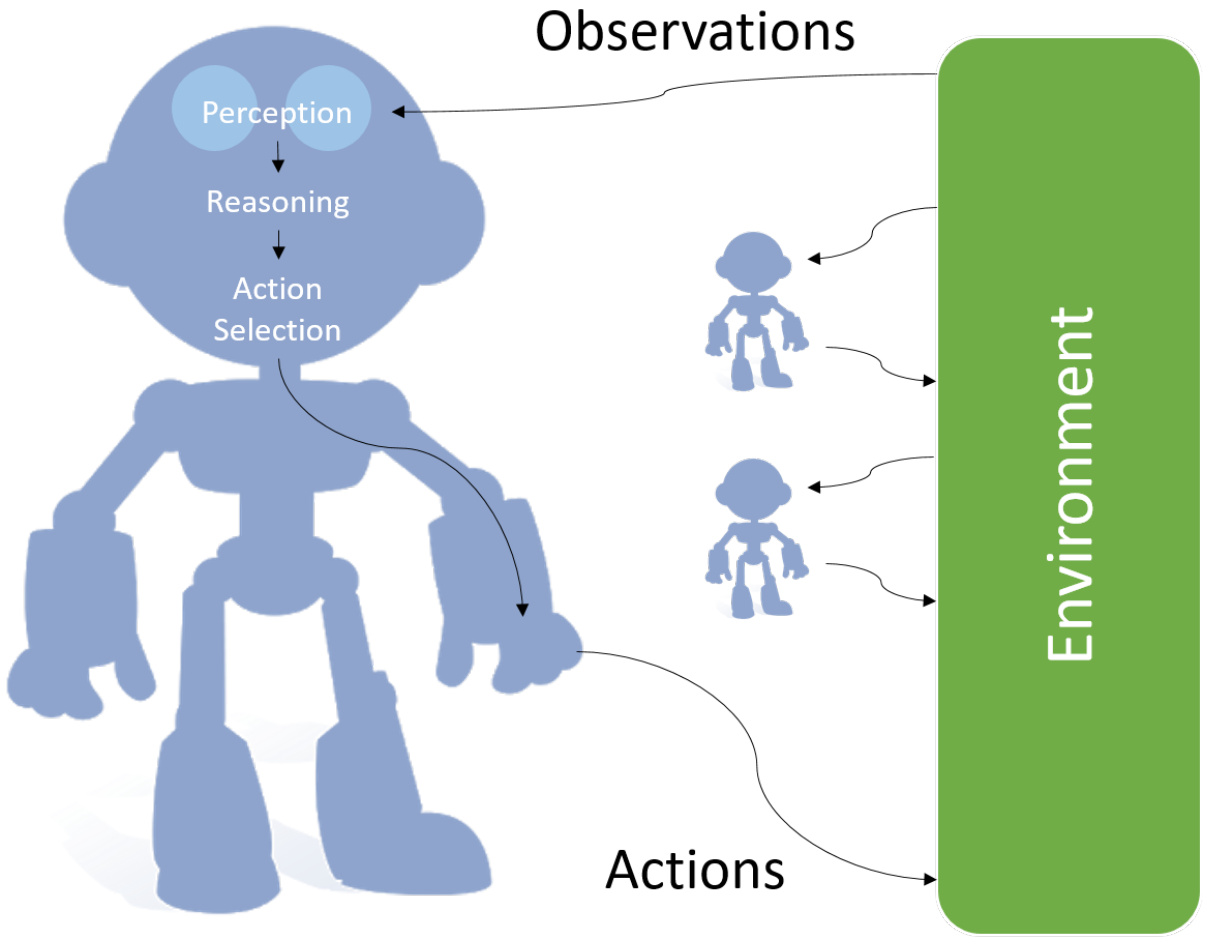
Imperial College London, UK

SafeAI 2019, Honolulu (HI)

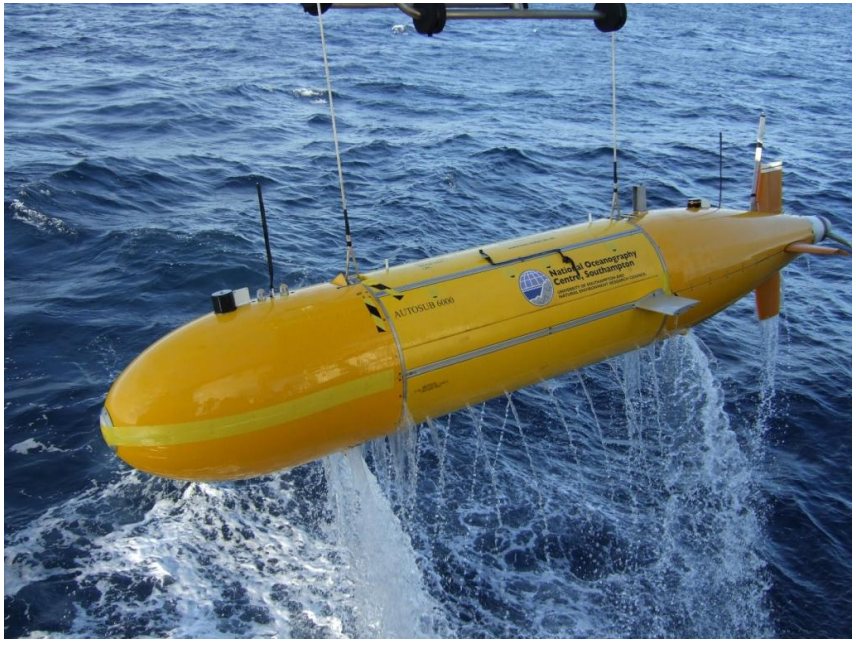
Safe and Trustworthy AI

- Verification of MAS: assessing whether a MAS is correct with respect to its specifications: $M_S \models \phi$.
- Specifications formally expressed in temporal, epistemic, BDI, languages expressing strategic interplay.
- Considerable amount of work from 2000, both theoretical investigations (complexity, etc.) and practical algorithms.
- Conquering large state spaces (bounded model checking, abstraction, symmetry reduction).
- Implementations including MCMAS (Imperial), MCK (UNSW), Verics (Warsaw).
- Applications in robotics, services, security, etc.

Multi-agent Systems



Assessing Correctness and Resilience of the Autsub6000



The NOC Autosub6000.

Weight: 5tons; Length: 5m;

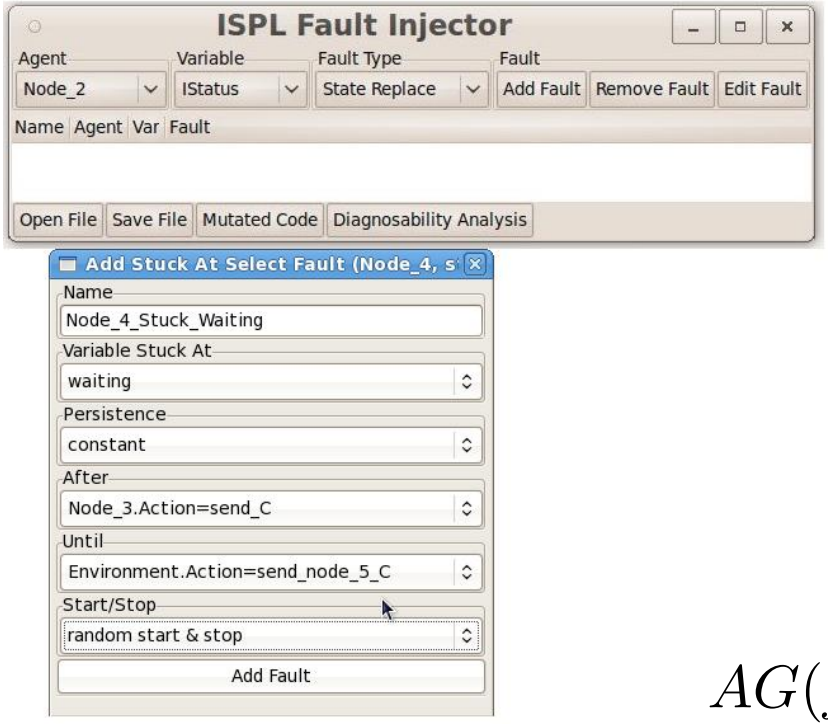
Range: 1000km; Max Depth: 6,000m;

Endur: 10days.

- Engineering design translated and discretised from Simulink/Stateflow into ISPL.
- Different sub-modules encoded as different cooperating agents.
- Several missions verified before deployment.
- **Key concern:** not just correctness but assessing fault-tolerance via fault-injection.

Reasoning About Fault-Tolerance of an AUV

From $M_S \models \phi_P$ to $M_S^* \models_S \phi'_P$, where M^* is a mutated model admitting faulty behaviour and ϕ'_P the specification of interest.



Recoverability on M_S^*

$$AG(fault \rightarrow \langle\langle \Gamma \rangle\rangle F \neg fault)$$

Diagnosability specs on M_S^*

$$AG(fault \rightarrow AFK_i fault)$$

$$AG(D_\Gamma fault \rightarrow AFC_\Gamma fault)$$

$$AG(fault \rightarrow AFK_i \bigvee_j fault_j \wedge \neg \bigwedge_j K_i fault_j)$$

The ISPL fault-injector compiler.

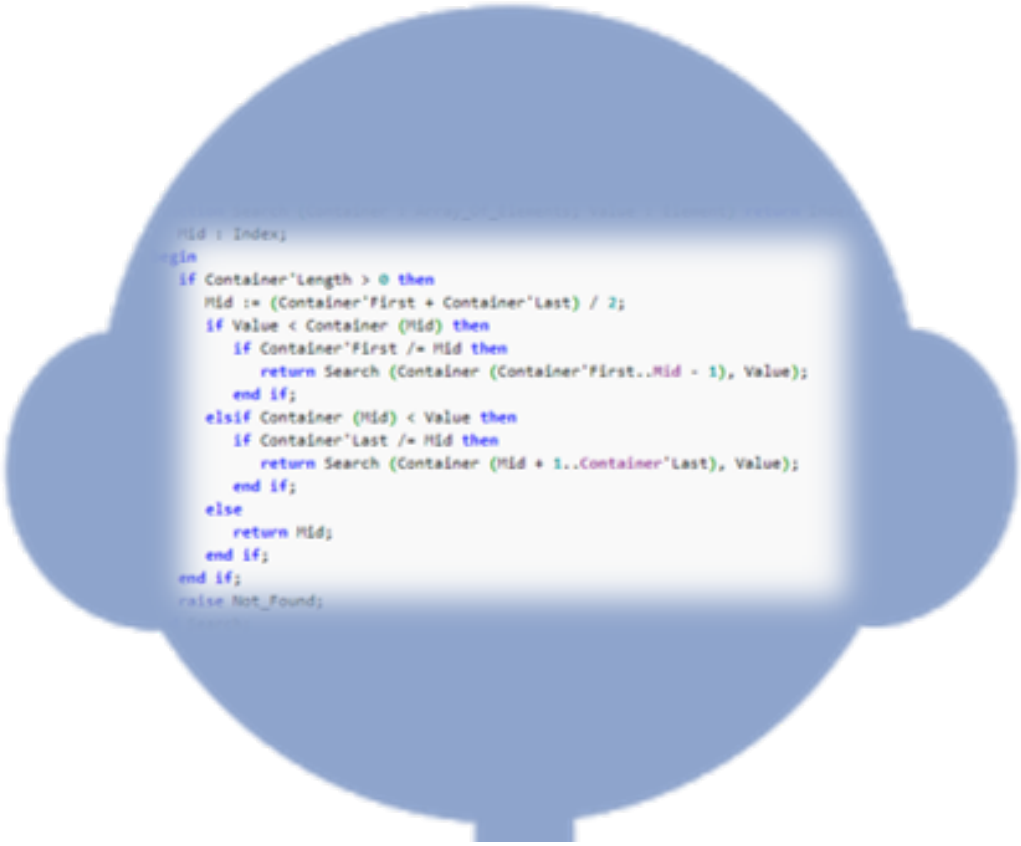
Problem solved?

Not quite:

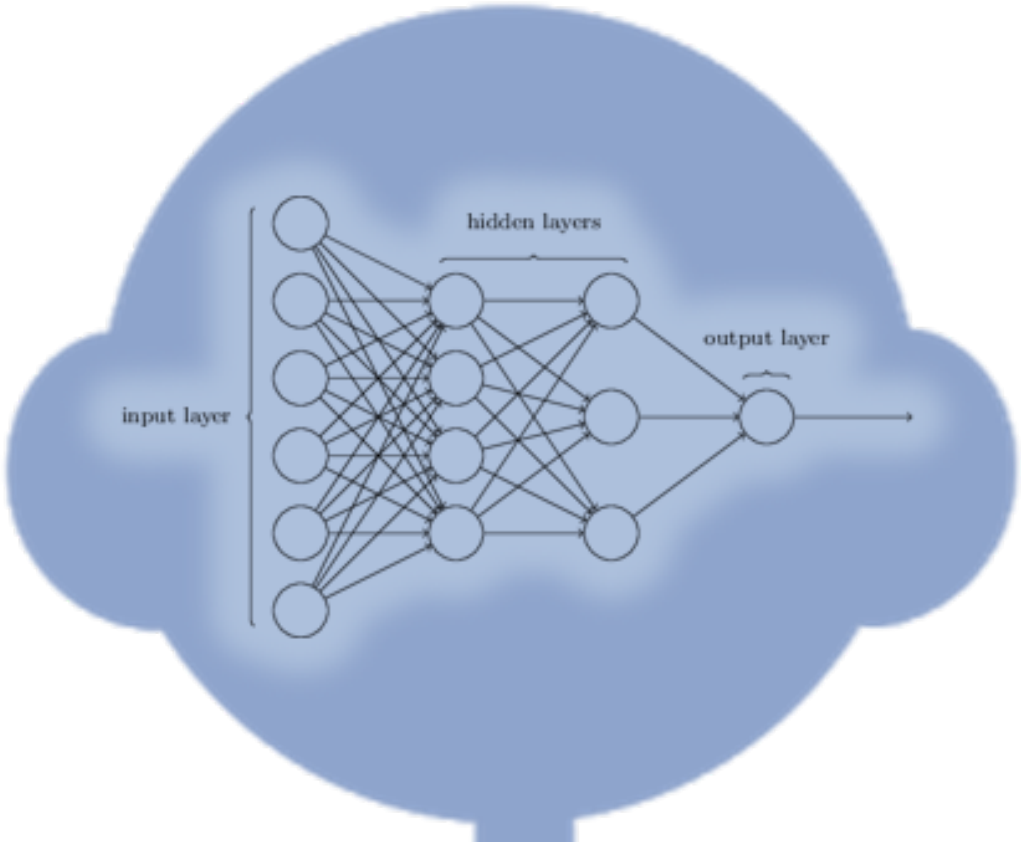
- Infinite state spaces.
- Expressive specifications.
- Parameterised systems, parameterised environments.
- Code verification.

Digging deeper, there is one **key** assumption in the above.

Key Assumption



Neural Agents



Paradigm shift. What are the implications of this?

Consequences?

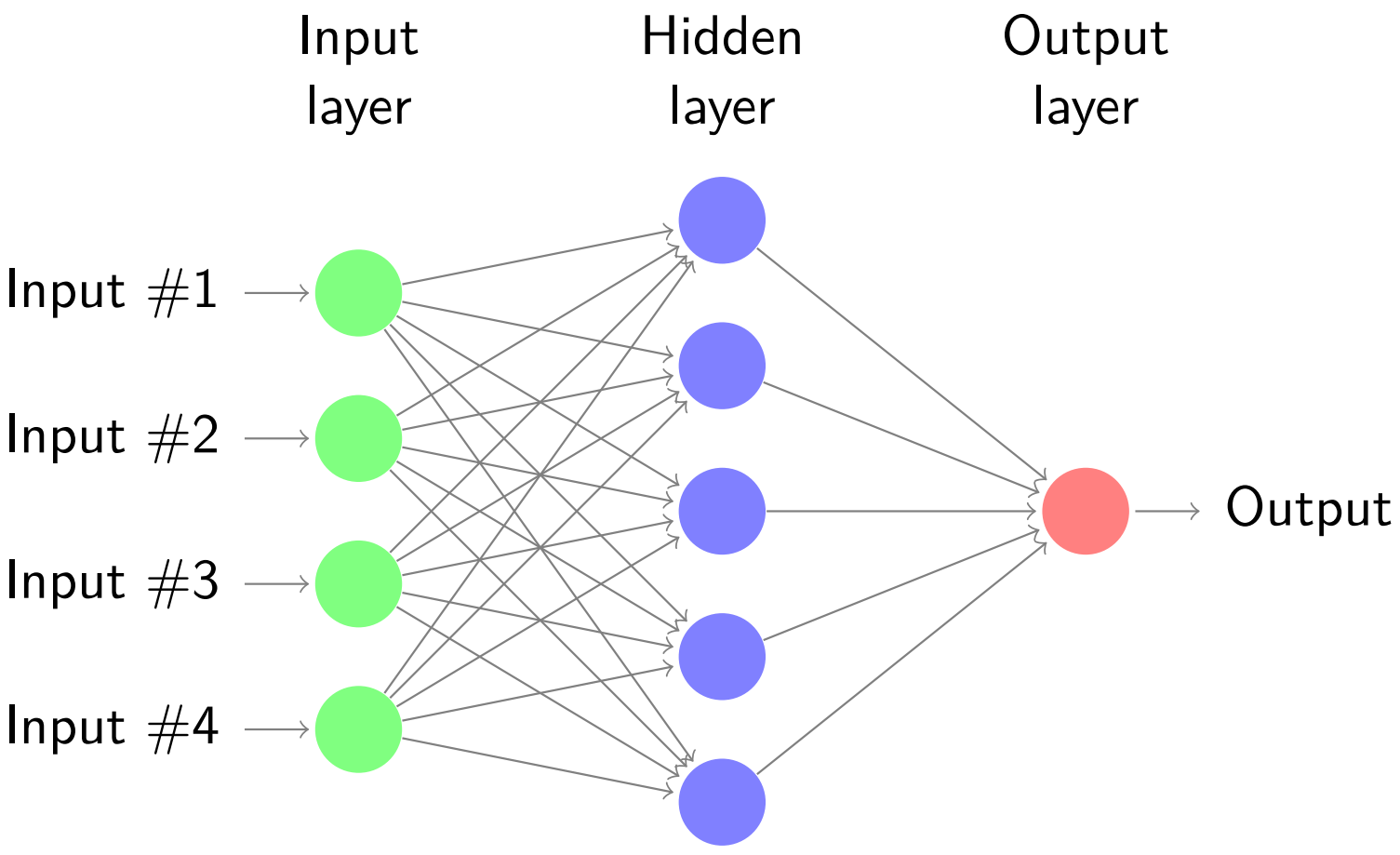
New methods required

Clear and urgent need of verification methods supporting agents in which some their components are synthesised via machine-learning.

This paper (Proc. of KR 2018)

- We put forward a notion of Neural Agent-Environment System to represent ML-based agents interacting with an environment.
- We define and study various verification problems for these systems.
- We define a method to solve said verification problems.
- We present an implementation and report experimental results.

Background: Deep Neural Networks



Background
Deep Neural Networks

- Deep neural networks (DNNs) consist of an input layer, output layer, and multiple *hidden* layers.
- Each layer contains multiple nodes, each connected to nodes from the preceding layer via a set of *weights*, which are determined during a *training* phase.
- Computation happens per layer where values are *fed forward* to the successive layers, eventually resulting in the final output values.

Various Neural Activation Functions

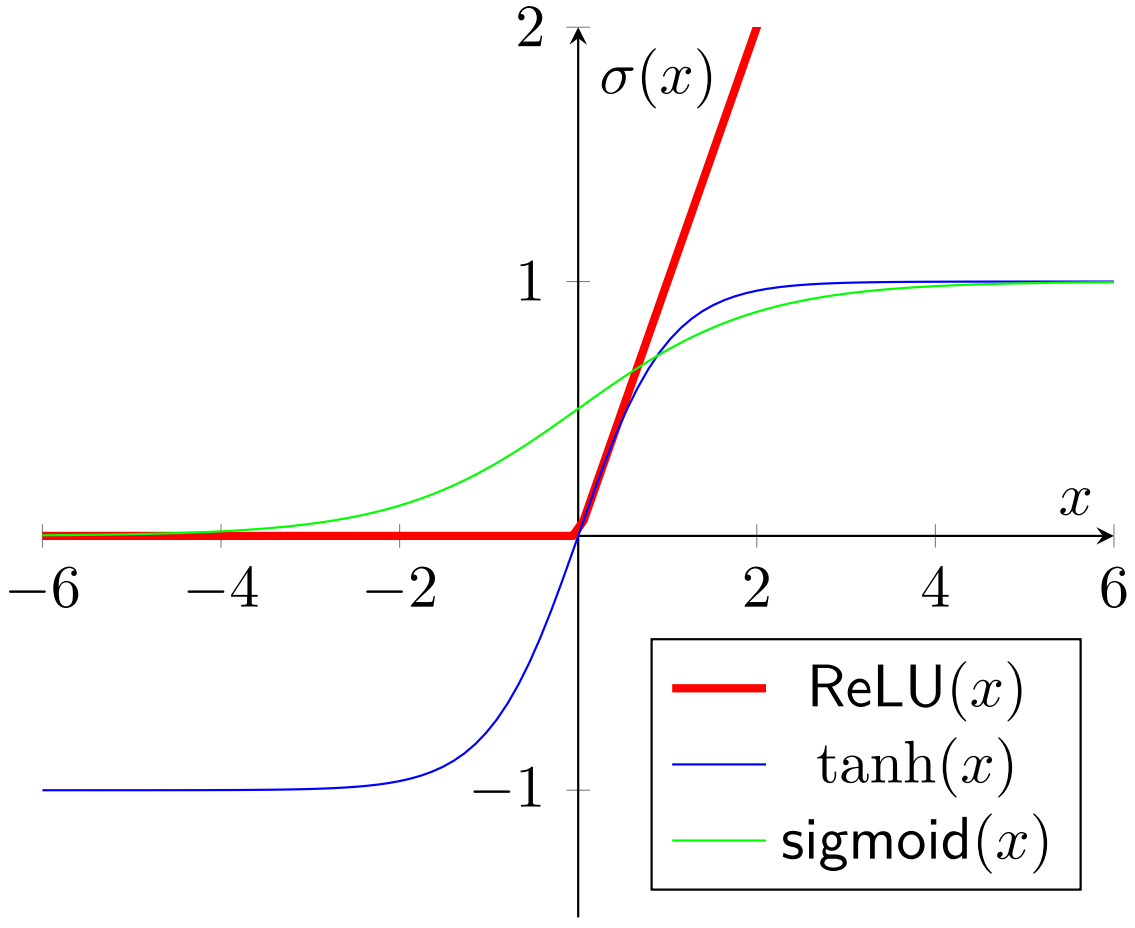


Figure: Common DNN activation functions

Neural Agent-Environment Systems: Intuition

- Rather than standard PL, (memoryless) agents run NNs.
- NNs for the agents define a mapping from observations to actions.
- NNs are all feed-forward ReLUs.
- Environment is linearly definable or approximated as such.

Neural Agents-Environment Systems: Formalisation

Definition (Agent-environment system)

An *agent-environment system* is a tuple $AES = (E, Agt, I)$ where:

- $E = (S, t_E)$ is an environment with corresponding state space S and transition function $t_E : S \times Act \to S$,
- Agt is a memoryless agent with corresponding action function $act : S \to Act$,
- $I \subseteq S$ is a set of initial states for the environment.

Since agents are memoryless, paths are sequences of env states determined by the transition function $t_E(s, (act(s)))$ from an initial state.
We assume linearly definable AES (both t_E and I).

Verification problems

- SSR: Single Step State Reachability: for a given O, is there an $i \in I$ s.t. $t^1(i) \in O$?
- MSR: Multi Step State Reachability: for a given O and $m \in N$, is there an $i \in I$ s.t. $t^m(i) \in O$?
- MAE: Multi Step Action Executability: for a given set of agent actions A and $m \in N$, is there an $i \in I$ s.t. $act(t^m(i)) \in A$?
- ASR: Arbitrary Step Reachability: for a given O, is there an $n \in N$ and an $i \in I$ s.t. $t^n(i) \in O$?

Mixed-Integer Linear Programs

Definition (Linear Programming)

- A function $f(x_1, \ldots, x_n)$ is said to be *linear* if for some $c \in \mathbb{R}^N$, we have $f(x_1, \ldots, x_n) = \sum_{i=1}^{N} c_i x_i$.
- A *linear constraint* is an expression of the form $f(x_1, \ldots, x_n) = b$, $f(x_1, \ldots, x_n) \leq b$ and $f(x_1, \ldots, x_n) \geq b$ where $f(x_1, \ldots, x_n)$ is a linear function.

Recall that a linear programming problem involves maximising a linear objective function subject to a set of linear constraints on the variables.

Definition (Mixed Integer Linear Programs)

A *mixed integer linear program* (MILP) is a linear programming problem on **real**, **binary** or **integer** decision variables.

MILP Encoding for Deep ReLU FF (Maganti, L.)

For a layer i with input vector $\bar{x}^{(i-1)}$ and output $\bar{x}^{(i)}$, associate the set of *linear constraints encoding layer* i:

$$C_i = \{\bar{x}_j^{(i)} \geq W_j^{(i)}\bar{x}^{(i-1)} + b_j^{(i)}, \bar{x}_j^{(i)} \leq W_j^{(i)}\bar{x}^{(i-1)} + b_j^{(i)} + M\bar{\delta}_j^{(i)},$$
$$\bar{x}_j^{(i)} \geq 0, \bar{x}_j^{(i)} \leq M(1 - \bar{\delta}_j^{(i)}) \mid j = 1...|L^{(i)}|\}$$

where M a "sufficiently large" constant, and $\bar{\delta}_j$ is a vector of binary variables.

MILP Encoding for Deep ReLU FF (Maganti, L., 2017)

ReLU activation function

$$\bar{x}_j^{(i)} = \max\left(0, W_j^{(i)}\bar{x}^{(i-1)} + b_j^{(i)}\right),\ j = 1 \cdots |L^{(i)}|$$

- Active phase: $\bar{x}_j^{(i)} = W_j^{(i)}\bar{x}^{(i-1)} + b_j^{(i)}$ (set $\bar{\delta}_j^{(i)} = 0$)
- Inactive phase: $\bar{x}_j^{(i)} = 0$ (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:

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

Equivalence between MILP encoding and NN computation

Consider the MILP problem P_{NN} defined by $obj_{NN} = 0$ and $C_{NN} = \cup_{i=2}^{m} C_i$, where m is the depth of NN.

Theorem

P is feasible for $\overline{x}^{(1)} = \overline{x}, \overline{x}^{(m)} = \overline{y}$ iff $f_{NN}(\overline{x}) = \overline{y}$.

This gives a procedure for computing one step computation for a deep ReLU NN.

Solving SSR, MSR, MAE via MILP

Theorem

Each decision problem in $\{SSR, MSR, MAE\}$ is fully characterised by its corresponding linear problem in $\{P_{SSR}, P_{MSR}, P_{MAE}\}$ defined on appropriate sets of constraints.

SSR, MSR, MAE can be solved via MILP by considering the corresponding linear programs.

Theorem

SSR, MSR, MAE are NP-complete.

Solving Arbitrary Step Reachability

- Consider MSR on i;
- If SAT, stop with output i;
- Compute all reachable points via MILP; if fixed point, stop;
- MSR($i+1$), etc.

Partial decision problem (complete under finite number of states).

NSverify

- Experimental toolkit solving SSR, MSR, MAE, ASR decision problems.
- Input: agent, environment, decision problem. Output: associated MILP problem.
- Fed to Gurobi 7.5.1 to solve them.
- If result is SAT, a trace is shown.

Example: OpenAI Pendulum I

Example (Pendulum)

OpenAI Gym [Brockman et. al, '16] task PENDULUM:

- System composed of a pendulum and an agent which can apply a force to the pendulum.
- The agent can observe the current angle of the pendulum (where an angle of zero indicates that it is perfectly vertical) along with the pendulum's angular velocity.
- Agent chooses a **torque** to be applied to the pendulum.
- Get the pendulum to an upright position and maintain it there.

Pendulum-v0 from OpenAI Gym

Agent observing the angle and applying a torque to keep it vertical.

- Encoded as a NAES: agent input-output, env state and transitions.
- Both agent and env learned from data (since env is non linear).

Single Step Reachability: NSVerify found several bugs in the synthesised controller agent, e.g., the agent would apply the torque incorrectly in some situations.

Multi-Step Reachability: Results

Results on the PENDULUM [Open AI, '18] dataset

Check $\theta_f \geq \varepsilon$ after n steps for different values of ε and n.

		ε			
		$\pi/70$	$\pi/100$	$\pi/200$	$\pi/500$
n	1	0.06s	0.06s	0.06s	0.06s
	2	0.26s	0.16s	0.16s	0.16s
	3	0.68s	1.20s	0.35s	0.34s
	4	1.54s	2.17s	1.69s	1.46s
	5	8.19s	8.26s	7.42s	3.01s
	6	20.29s	16.91s	17.06s	18.44s
	7	38.49s	32.51s	69.95s	29.11s
	8	77.42s	83.29s	149.81s	99.77s

Greyed areas denote UNSAT, hence correctness.

For n steps the problem contains $1214n$ constraints on $973n$ vars.

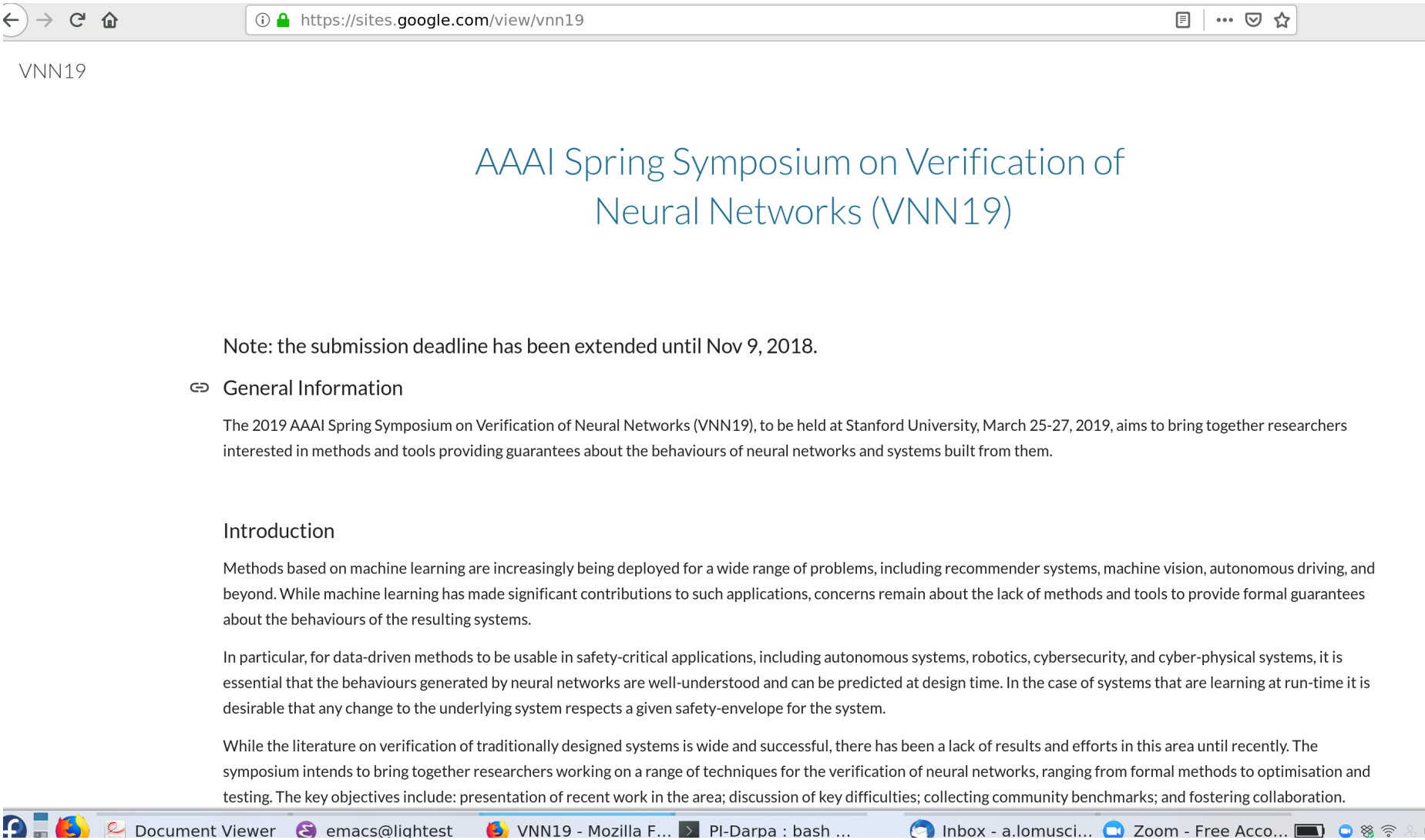
NSVerify comparison for "Pure" NN reachability

Tool	Success rate	Average runtime	Number of wins
NSVerify	62.96%	2106.8s	16
Reluplex	53.08%	2135.5s	4
Mip	69.14%	2430.2s	7
BaB	29.63%	5212.5s	5
Planet	64.20%	2807.9s	29

Table: Results on the TwinStream dataset.

Recurrent Neural Agent-Environment Systems (AAAI19)

- Rather than memoryless FFNN agents, we consider stateful agents running ReLU-based RNNs.
- RNNs define a mapping from a sequence of observations to actions (rather than single observation).
- Environment is linearly definable or approximated as such.
- Verification problem solved by unravelling RNNs into FFNNs.
- Not just reachability but bounded version of LTL.


https://sites.google.com/view/vnn19
VNN19
AAAI Spring Symposium on Verification of Neural Networks (VNN19)
Note: the submission deadline has been extended until Nov 9, 2018.
General Information
The 2019 AAAI Spring Symposium on Verification of Neural Networks (VNN19), to be held at Stanford University, March 25-27, 2019, aims to bring together researchers interested in methods and tools providing guarantees about the behaviours of neural networks and systems built from them.
Introduction
Methods based on machine learning are increasingly being deployed for a wide range of problems, including recommender systems, machine vision, autonomous driving, and beyond. While machine learning has made significant contributions to such applications, concerns remain about the lack of methods and tools to provide formal guarantees about the behaviours of the resulting systems.
In particular, for data-driven methods to be usable in safety-critical applications, including autonomous systems, robotics, cybersecurity, and cyber-physical systems, it is essential that the behaviours generated by neural networks are well-understood and can be predicted at design time. In the case of systems that are learning at run-time it is desirable that any change to the underlying system respects a given safety-envelope for the system.
While the literature on verification of traditionally designed systems is wide and successful, there has been a lack of results and efforts in this area until recently. The symposium intends to bring together researchers working on a range of techniques for the verification of neural networks, ranging from formal methods to optimisation and testing. The key objectives include: presentation of recent work in the area; discussion of key difficulties; collecting community benchmarks; and fostering collaboration.
Document Viewer
emacs@lightest
VNN19 - Mozilla F...
PI-Darpa : bash ...
Inbox - a.lomusci...
Zoom - Free Acco...

Conclusions

- Increased attention to Safe and Trustworthy AI.
- First approach on verification of closed-loop “neural agents”.
- Reachability a key property.
- Approach is independent of the underlying solver.
- Much more to do!

References

- A. Lomuscio, L. Maganti. An approach to reachability analysis for feed-forward ReLU neural Networks. arXiv preprint. arXiv:1706.07351 [cs.AI]. 2017
- M. Akintunde, A. Lomuscio, L. Maganti, E. Pirovano. Reachability Analysis for Neural Agent-Environment Systems. Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning (KR18). Tempe, Arizona. AAAI Press. November 2018.
- M. Akintunde, A. Kevorchian, A. Lomuscio, E. Pirovano. Verification of RNN-Based Neural Agent-Environment Systems.Proceedings of the 33th AAAI Conference on Artificial Intelligence (AAAI19). Honolulu, HI, USA. AAAI Press. To appear.
- P. Kouvaros, A. Lomuscio. Verification of CNN-based perception systems. arXiv preprint. arXiv:1811.11373.