Reachability for neural-network control systems

Christian Schilling

AALBORG UNIVERSITY

AISoLA 2023

Neural networks

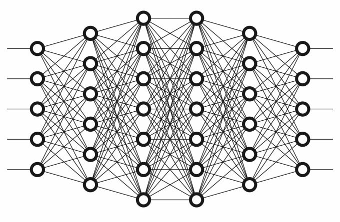
- Layer $\ell : \mathbb{R}^m \to \mathbb{R}^n$: affine map followed by activation function

$$\ell(x) = \alpha_\ell(Wx + b)$$

- Neural network $N : \mathbb{R}^m \to \mathbb{R}^n$: composition of k layers

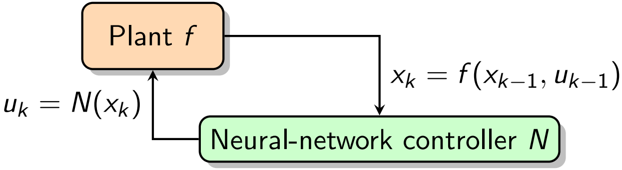
$$N(x) = (\ell_k \circ \cdots \circ \ell_1)(x)$$

- **ReLU NN** if $\alpha_{\ell_k} = \mathrm{id}$ and $\alpha_{\ell_i} = \rho$ for $i < k$, where $\mathrm{id}(y) = y$ and $\rho(y) = \max(y, 0)$ componentwise



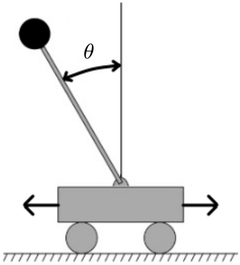
Neural-network control system

- Plant (or environment) can use discrete or continuous time
- Here we consider the discrete-time case

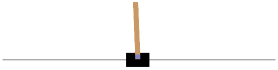


Example: Stabilization of a pole on a cart

- Continuous time



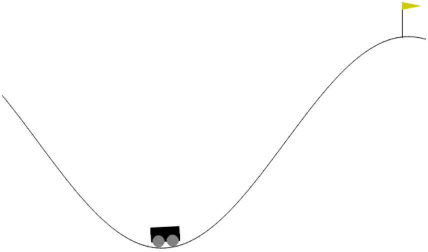
$$\dot{p} = v$$
$$\dot{\theta} = \omega$$
$$\dot{\omega} = \phi$$
$$\dot{v} = \psi - \frac{1}{22}\phi\cos(\theta)$$
$$\phi = \frac{9.8\sin(\theta) - \cos(\theta)\psi}{2/3 + 5/11\cos(\theta)^2}$$
$$\psi = \frac{10\boldsymbol{u} + 0.05\omega^2\sin(\theta)}{1.1}$$



Example: Car reaching top of the mountain

- Discrete time

$$v_{k+1} = v_k + (u-1)F - \cos(3p_k)g$$
$$p_{k+1} = p_k + v_{k+1}$$



Reachability problem

- Given a ReLU NN $N : \mathbb{R}^m \to \mathbb{R}^n$, a plant $f : \mathbb{R}^n \to \mathbb{R}^m$, an initial valuation $x_0 \in \mathbb{R}^n$, and an output property $\phi_{\text{out}} \subseteq \mathbb{R}^n$
- We define the **reachability problem** as determining whether there exists $k \in \mathbb{N}$ such that

$$(f \circ N)^k(x_0) \in \phi_{\text{out}}$$

Reachability problem

- Given a ReLU NN $N : \mathbb{R}^m \to \mathbb{R}^n$, a plant $f : \mathbb{R}^n \to \mathbb{R}^m$, an initial valuation $x_0 \in \mathbb{R}^n$, and an output property $\phi_{\text{out}} \subseteq \mathbb{R}^n$
- We define the **reachability problem** as determining whether there exists $k \in \mathbb{N}$ such that

$$(f \circ N)^k(x_0) \in \phi_{\text{out}}$$

Theorem

The **reachability problem** for ReLU NNs is undecidable

Prior results

- For any computable function one can build a processor net[1] with external inputs,

$$x(0) = 0, \quad x(k+1) = \alpha(Ax(k) + bu(k) + c),$$

and truncated ReLU

$$\alpha(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases}$$

- Encoding of Turing machine in **ReLU recurrent NN**[2]

[1] H. T. Siegelmann and E. D. Sontag. *Applied Mathematics Letters* (1991).

[2] H. Hyötyniemi. *STeP* (1996).

Two-counter machines

- We consider a **two-counter machine** (2CM)[1] with two counters c_x and c_y and instruction set $\texttt{INC}(c_z)$, $\texttt{DEC}(c_z)$, and $\texttt{JZ}(c_z, j')$, for $z \in \{x, y\}$
- Define $c_1 \ominus 1 = \begin{cases} c_1 - 1 & c_1 > 0 \\ c_1 & c_1 = 0 \end{cases}$
- We write z for the value currently stored in counter c_z
- Each instruction I comes with a position j in the program, which we write "$j : I$"
- Initial value of counters is $x = y = 0$
- Program starts at instruction 1

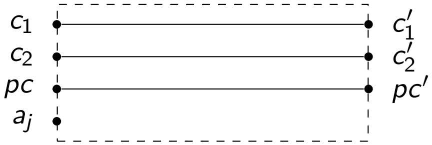
[1] J. C. Shepherdson and H. E. Sturgis. *J. ACM* (1963).

Reduction

- Given: 2CM with k instructions
- Let $m = n = 3$ and $f(x, u) = u$
- NN inputs (and outputs) v_1, v_2, v_3 represent values of counters, x and y, and program counter pc
- Idea: build a gadget for each instruction in the 2CM
- NN operates over integers

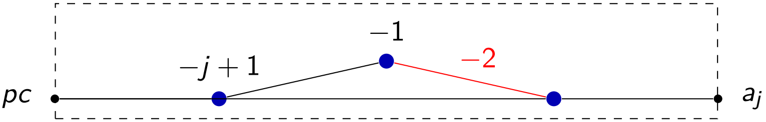
Gadget structure

- All gadgets will be executed in parallel in one iteration of the NN, but only one of them (determined by the program counter) will actually perform a computation
- The other gadgets will just compute the identity function for each counter as well as the program counter
- In the end we need to subtract $(k-1) \cdot v$ from each input v
- We control gadget j with an auxiliary bit a_j



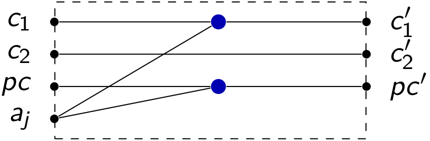
Auxiliary gadget

- Thick blue dots are (ReLU) neurons
- We omit weight 1 and bias 0
- We omit auxiliary neurons to preserve structure



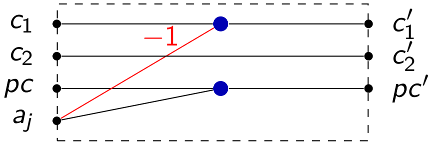
$$a_j = \begin{cases} 1 & pc = j \\ 0 & pc \neq j \end{cases}$$

Gadget for j : $\mathtt{INC}(c_1)$



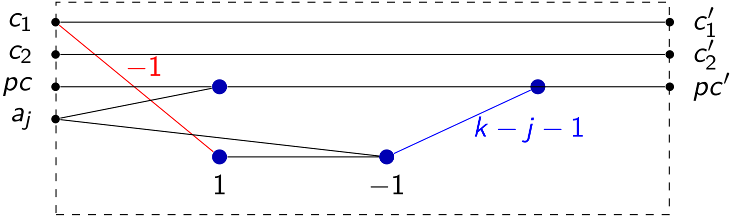
	c_1'	c_2'	pc'
$pc = j$	$c_1 + 1$	c_2	$pc + 1$
$pc \neq j$	c_1	c_2	pc

Gadget for j : $\mathtt{DEC}(c_1)$



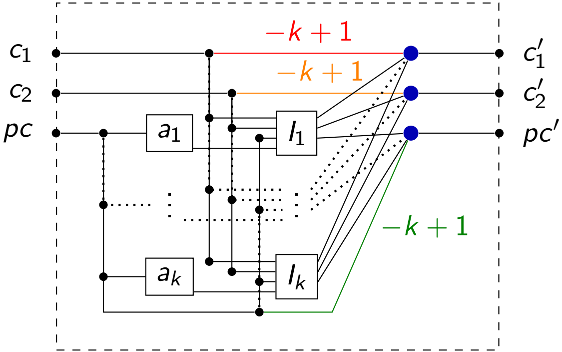
	c_1'	c_2'	pc'
$pc = j$	$c_1 \ominus 1$	c_2	$pc + 1$
$pc \neq j$	c_1	c_2	pc

Gadget for j : $\mathrm{JZ}(c_1, a_k)$



	c_1'	c_2'	pc'
$pc = j, c_1 = 0$	c_1	c_2	k
$pc = j, c_1 \neq 0$	c_1	c_2	$pc + 1$
$pc \neq j$	c_1	c_2	pc

Putting everything together



Last step

- Initial valuation $x_0 = (0, 0, 1)$
- Output property $\phi_{\text{out}} = \{(x, y, z) \mid z = k + 1\}$ (reaching the end of the program)
- Depth of NN: $7 = 3\ (a_j) + 3\ (\text{gadgets}) + 1\ (\text{correction})$

Corollary

The **reachability problem** remains undecidable with fixed depth 7 and a hyperplanar output property $x = v$ for some output neuron x and constant $v \in \mathbb{N}$

Conclusion

- **Reachability problem** for **ReLU NN control systems** is **undecidable**, even in the very restricted setting $f(x, u) = u$
- Similar result for decision-tree control systems

Theorem[1]

The **reachability problem** assuming $f(x, u) = u$ is **undecidable**, and **PSPACE-complete** for a bounded number of steps

- Open question
  - Complexity for **ReLU NN control systems** with bounded number of steps

[1] C. Schilling, A. Lukina, E. Demirović, and K. G. Larsen. *NeurIPS*. 2023.