An Abstraction-Based Framework for Neural Network Verification

Yizhak Yisrael Elboher, Justin Gottschlich, and Guy Katz, CAV 2020

Kumar Madhukar

Formal Methods Update Meeting 2022

Indian Institute of Technology Delhi

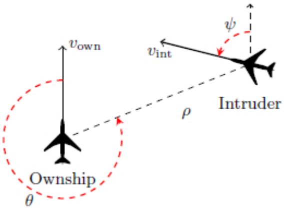
5 July 2022

Deep Neural Networks

- Deep Neural Networks (DNNs) are everywhere
- they are artifacts produced by Machine Learning (ML)
  - an ML algorithm *generalizes* a set of examples into a DNN
  - behave correctly for previously-unseen inputs
- image/speech recognition, game playing, NLP, etc.
- can be easier to create than handcrafted software
- effective means to implement complex software systems

Airborne Collision Avoidance System

- in response to midair collisions between commercial aircrafts
- used to be a lookup table (of size 2GB), mapping sensor measurements to advisories



- replaced by DNNs (less than 3MB of memory)
- continuous in nature, better than (discrete) lookup tables
- necessitates formal verification

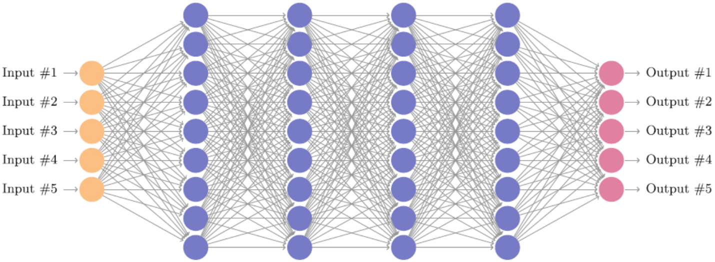
Neural Network Verification

- small changes to correctly handled inputs may lead to unexpected (and erroneous) behaviors
- testing cannot prove inexistence of faulty behaviors
- there are techniques that can automatically prove that a DNN satisfies a prescribed property
- hard problem; becomes exponentially more difficult as network size increases
- paper's contribution: an abstraction-refinement technique

A well-known story in formal verification

- replace the DNN $\mathcal{N}$ by a "smaller" (*abstract*) network $\overline{\mathcal{N}}$
- verify $\overline{\mathcal{N}}$; by construction, if $\overline{\mathcal{N}}$ meets the spec, so does $\mathcal{N}$
- if $\overline{\mathcal{N}}$ fails to meet the spec, there must be counterexample x
- if x is actual, $\mathcal{N}$ violates the spec
- else refine $\overline{\mathcal{N}}$ (little more accurate, and "larger")
- done using the spurious counterexample x
  (Counterexample-Guided Abstraction Refinement, or CEGAR)

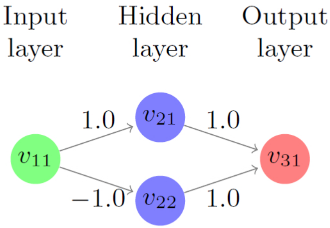
Background: Neural Networks



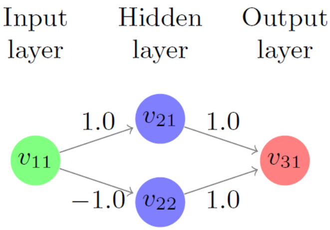
- feedforward network
  - edges have weights, neurons have activation function
- evaluate a neuron: compute weighted sum, and apply activation function
- $\text{ReLU}(x) = \max(0,x)$, called Rectified Linear Unit

Verification

- precondition $\mathcal{P}$, postcondition $\mathcal{Q}$, network $\mathcal{N}$
- is there an input x that satisfies $\mathcal{P}(x)$ and $\mathcal{Q}(y)$, where $y = \mathcal{N}(x)$



Verification



- is the output (v31) always equal to the input (v11)?
- is it possible that v11 $\in$ [0,1] and v31 $\in$ [0.5, 1]
- is v11 always equal to v31 for non-negative inputs?

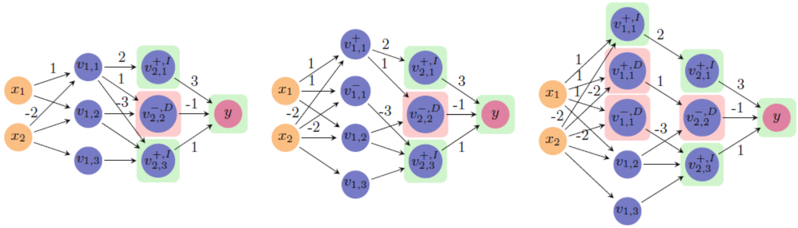
Verification

- precondition $\mathcal{P}$, postcondition $\mathcal{Q}$, network $\mathcal{N}$
- is there an input x that satisfies $\mathcal{P}(x)$ and $\mathcal{Q}(y)$, where $y = \mathcal{N}(x)$
- assumptions made in this paper:
  - (on $\mathcal{N}$) - only ReLU activation functions; single output node
  - (on $\mathcal{P}$) - conjunctions of linear constraints on input values
  - (on $\mathcal{Q}$) - $y > c$, for a given constant c
- not as limiting as it may seem (let us come back to this in the end)

Abstraction

- transform the neural network $\mathcal{N}$ into $\overline{\mathcal{N}}$, such that $\mathcal{N}(x) \leq \overline{\mathcal{N}}(x)$, for every input x
- if abstract is safe ($\overline{\mathcal{N}}(x) \leq c$), then so is the concrete ($\mathcal{N}(x) \leq c$)
- abstraction-refinement: merging neurons (and then splitting back)
- but not on $\mathcal{N}$ (on an equivalent network $\mathcal{N}''$)

$\mathcal{N} \rightarrow \mathcal{N}' \rightarrow \mathcal{N}''$ (all equivalent)

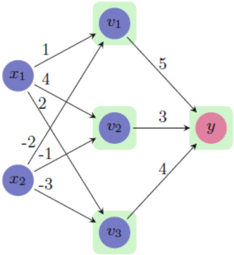


- every hidden neuron should either be `pos` or `neg`
- based on weights of outgoing edges; split if needed ($\mathcal{N}'$)
- also, every neuron must be `inc` or `dec`; split if needed
- depending on whether increasing (or decreasing) its value results in an increased output (traversing backwards)

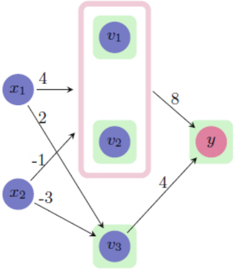
The abstract operator

- merges a pair of neurons; can be done multiple times
- merge only if the `pos`/`neg` and `inc`/`dec` attributes are same
- for the (`pos, inc`) and (`neg, inc`) case
  - take max of incoming, and sum of outgoing
- for the (`pos, dec`) and (`neg, dec`) case
  - take min of incoming, and sum of outgoing
- intuitively, the new node contributes more to the output (than the two original nodes)

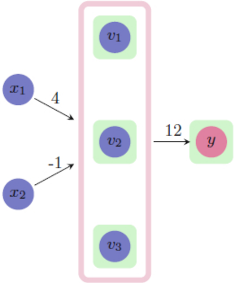
An example



$$y = 5R(x_1 - 2x_2) + 3R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$$



$$y = 8R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$$



$$y = 12R(4x_1 - x_2)$$

- abstraction is independent of the order in which it was done

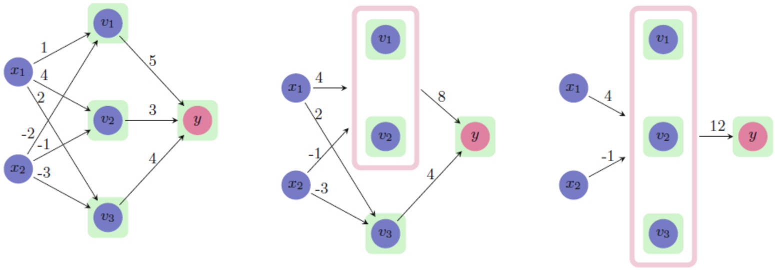
The need to refine

- of course, if the abstraction is too coarse
- suppose $\mathcal{N}(x_0) = 3$, $\overline{\mathcal{N}}(x_0) = 8$, and the property is $\overline{\mathcal{N}}(x) > 6$
- need to refine $\overline{\mathcal{N}}$ into $\overline{\mathcal{N}}'$, such that for every x, $\mathcal{N}(x) \leq \overline{\mathcal{N}}'(x) \leq \overline{\mathcal{N}}(x)$
- refine picks a concrete neuron from an abstract neuron, and puts it back in the network

More about the abstraction

- apply *abstraction to saturation* (to at most 4 neurons in every hidden layer)
- can be controlled based on certain heuristics
- inaccuracies by caused by the max and min operators
- merge neurons that approximate least; split one that restores the most

Merging heuristics

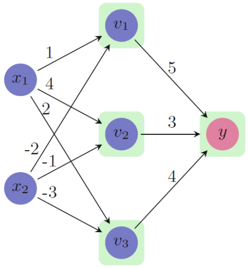


$y = 5R(x_1 - 2x_2) + 3R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$

$y = 8R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$

$y = 12R(4x_1 - x_2)$

- merge: maximal value of $|a - b|$ (over all incoming edges with weights a and b) is minimal
- the new edge is "closest" to the replaced ones (reducing a neuron anyway!)



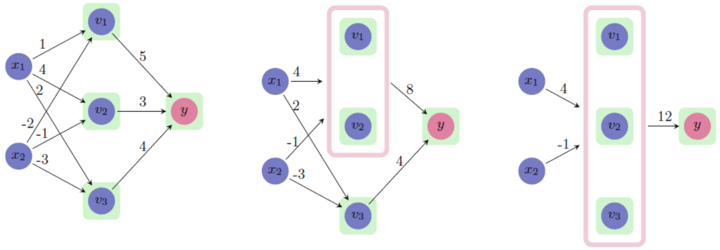
- merging (v_1, v_2), the (a, b) pairs are: (1,4), (-2, -1)
- $\max(\ |1-4|\ , \big|-2-(-1)\big|\ ) = 3$

- merging (v_1, v_3), the (a, b) pairs are: (1,2), (-2, -3)
- $\max(\ |1-2|\ , \big|-2-(-3)\big|\ ) = 1$

- merging (v_2, v_3), the (a, b) pairs are: (4,2), (-1, -3)
- $\max(\ |1-2|\ , \big|-2-(-3)\big|\ ) = 2$

- merge (v_1, v_3) first

Splitting heuristics

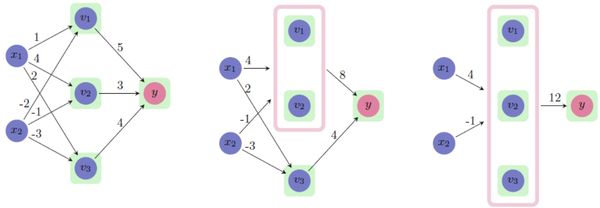


$$y = 5R(x_1 - 2x_2) + 3R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$$

$$y = 8R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$$

$$y = 12R(4x_1 - x_2)$$

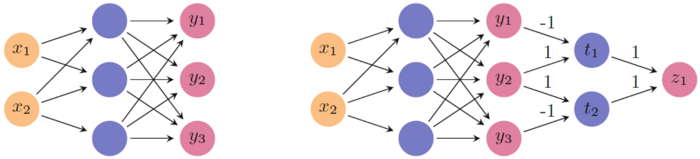
- split: v from $\overline{v}$, by considering
  - edge-weight difference between v and $\overline{v}$
  - difference between $v(x)$ and $\overline{v}(x)$, for the counterexample x



$y = 5R(x_1 - 2x_2) + 3R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$ $\quad$ $y = 8R(4x_1 - x_2) + 4R(2x_1 - 3x_2)$ $\quad$ $y = 12R(4x_1 - x_2)$

- consider the counterexample $(x_1 = 1, x_2 = 0)$
- original neurons' evaluation: $(v_1 = 1, v_2 = 4, v_3 = 2)$
- abstract neuron's evaluation: $(\overline{v} = 4)$
- wt. diff. (between v_1 and $\overline{v}$) for in-edge from x_1, x_2: 3, 1
- wt. diff. (between v_2 and $\overline{v}$) for in-edge from x_1, x_2: 0, 0
- wt. diff. (between v_3 and $\overline{v}$) for in-edge from x_1, x_2: 2, 2
- remove v_1, (wt. diff $*$ val. diff.) is largest: (9, 0, 4)

Reducing a complex property (in the desired form)



- consider the property $(y_2 > y_1) \vee (y_2 > y_3)$
- encoded by adding neurons t_1, t_2, and z_1
- $t_1 = \max(0, y_2 - y_1)$
- $t_2 = \max(0, y_2 - y_3)$
- $z_1 = t_1 + t_2$
- property: $z_1 > 0$ (*iff* $t_1 > 0 \vee t_2 > 0$)

Experiments

- 45 DNNs from ACAS
- input is a set of sensor readings (speed, direction, location, etc.)
- five output neurons - possible turning advisories (left, right, clear-of-conflict, etc.)
- each DNN has 300 hidden neurons, across 6 hidden layers (leading to 1200 neurons after the transformation)

Findings

- abstraction to saturation outperforms indicator-guided abstraction
- avg. 269 nodes were needed to prove (the original has 310)
- “simpler” queries may sometimes be better than smaller networks
- reconfirmed in another set of experiments: even though network size increased (to avg. 385, from 310), abstracted versions were easier to verify that the original
- even further reduction on adversarial robustness properties

Summary

- pre-processing DNNs can be very helpful
- merging based on semantic similarity has also been explored
- should be possible to do both
- would be good to identify not just the behavior, but also which neurons are important

Thank you!