Safety Assurance in Cyber-Physical Systems built with Learning-Enabled Components (LECs)

December 12, 2018

Taylor T. Johnson
taylor.johnson@vanderbilt.edu

VeriVITAL - The Verification and Validation for Intelligent and Trustworthy Autonomy Laboratory (http://www.VeriVITAL.com)

Institute for Software Integrated Systems (ISIS)
Electrical Engineering and Computer Science (EECS)
Cyber-Physical Systems (CPS)

All of examples are safety-critical CPS!

Can we bet our lives on autonomous CPS designs?
A team of undergraduate engineering students programmed different drone types to fly in formation. The 40 swarm drones are expected to be used in an upcoming CS class.

Juniors Stirling Carter & Austin Williams (L) and junior Yinghni Yang, Anissa Aledander (BE ’17) & sophomore Timothy Lang (R)

Faculty Mentor: Taylor T. Johnson
assistant professor of computer science, computer engineering, and electrical engineering

Electrical Engineering and Computer Science
Motivation: Perdix, Autonomous Distributed CPS

https://www.youtube.com/watch?v=bsKbGc9TUHc
Motivation: Chinese Boat Swarm, Autonomous Distributed CPS
(Formal) Verification and Validation (V&V) Challenge

Given system model $\mathcal{A}$ and property $P$, design *algorithm* that returns
$\mathcal{A}$ satisfies $P$ and give *proof*, or $\mathcal{A}$ violates $P$ and why (bug)

Engineering Grand Challenge

- Debugging & verification: $\sim50\%-75\%$ engineering cost [Beizer 1990]
- Expensive & life-threatening bugs: $\sim$\$60 billion/year [NIST 2002]
- Fundamental & foundational computationally hard: State-space explosion (“curse of dimensionality”) & undecidability
  - Roughly: V&V gets exponentially harder in the size of the system

$\mathcal{A}$ networked software interacting with physical world: *cyber-physical systems (CPS)*

$P$
- **Safety**: something bad *never* happens
- **Stability**: reach good state *eventually* and stay there
- **Assurance**: safety, stability, liveness, mission specs, other functional & non-functional specs (security, performance, ...)

$\mathcal{A} \models P$?

Yes: proof

No: bug
Challenges for Assurance of LECs

• Non-transparency
  • LECs encode information in a complex manner and it is hard for humans to reason about the encoding
  • Non-transparency is an obstacle to safety assurance because it is more difficult to develop confidence that the model is operating as intended

• Error rate
  • LECs typically exhibit some nonzero error rate
  • True error rate unknown and only estimates from statistical processes known

• Training based
  • Training dataset is necessarily incomplete
  • May under-represent safety critical cases

• Unpredictable behavior
  • Training based on non-convex optimization algorithms and may converge to local minima
  • Changing training dataset may change behaviors

• LECs can exhibit unique hazards
  • Adversarial examples (incorrect output for a given input that cannot be discovered at design time): whole field of adversarial machine learning
  • May be always possible to find adversarial examples
  • Perception of environment is a functionality that is difficult to specify (typically based on examples)

[ https://www.labsix.org ]
Are autonomous cars today safer than human drivers?

• Standard metric: fatalities per mile driven

• Humans in the US:
  • Drive >3 trillion miles (~1/2 a light year!!!) annually (2016)
    • https://www.afdc.energy.gov/data/10315
    • Globally: over a light year
  • Around 37,000 fatalities (2016)
    • http://www.iihs.org/iihs/topics/t/general-statistics/fatalityfacts/state-by-state-overview
    • Dividing: approximately 1 fatality per 85 million miles driven by humans

• Autonomous vehicles
  • In total across all manufacturers, have driven on the order of ~10 million miles total
    • Ideal conditions in general (good weather, etc.)
      • https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/disengagement_report_2017
      • https://www.wired.com/story/self-driving-cars-disengagement-reports/
      • https://medium.com/waymo/waymo-reaches-5-million-self-driven-miles-61fba590fafe
    • Autonomous vehicles: at least one fatality (and probably ~5-10)
      • Dividing: approximately 1 fatality per 1 to 10 million miles driven

• Humans today are 1-2 orders of magnitude safer than current autonomous vehicles
Closed-Loop CPS with LECs Verification Flow and Tools

- Plant models: hybrid automata, or networks thereof, represented in HyST/SpaceEx/CIF formats
- LEC and cyber models: for now, neural networks, represented in ONNX format
- Specifications: primarily safety properties for now, some reachability properties
- Verification: composed LEC and plant analysis
Plant Modeling & Verification
HyST: Hybrid Source Transformation and Translation Software Tool

¬ https://github.com/verivital/hyst
LEC Verification
nnv: Neural Network Verifier Software Tool

- Preliminary software tool now available
  - Matlab toolbox for verification of neural networks
  - Available at: https://github.com/verivital/nnv

- Additionally, translators for common neural network formats, as well as to several other custom inputs required by other LEC verification tools (e.g., ReLUplex, Sherlock, …) in our NNMT tool
  - Available at: https://github.com/verivital/nnmt

- Current support:
  - Feedforward neural networks with ReLUs, tanh, and other monotonic activation functions
  - Closed-loop systems with LECs

- Method: reachability analysis-based verification
- Dependencies: Matlab MPT toolbox (https://www.mpt3.org/)

LEC Example:
Reachable set reaches unsafe region ($y_1 \geq 5$), the FFNN is unsafe

Unsafe region
Given a feedforward neural network $F$ and an input set $\mathcal{X}$, the **output reachable set** of the neural network $F$ is defined as $\mathcal{Y} = \{y[L] \mid y[L] = F(x[0], x[0] \in \mathcal{X})\}$.
Reachable Set Computation

Verification problem: Will neural network system $A$ satisfy or violate $P$?
ReLU (Rectified Linear Unit) Neural Network

For single neuron:

\[ y_j = f \left( \sum_{i=1}^{n} \omega_i x_i + \theta_i \right) = \max(0, \sum_{i=1}^{n} \omega_i x_i + \theta_i) \]

For single layer:

\[ x \rightarrow y = \max(0, Wx + \theta) \]

Theorem: For ReLU neural networks, if input set is a union of polytopes, then output sets of each layer are union of polytopes.

Input set:

\[ \mathcal{X}^{[0]} = \bigcup_{s=1}^{N_0} \mathcal{X}^{[0]}_s \]

\[ \mathcal{X}^{[0]}_s \triangleq \left\{ x^{[0]} \mid A^{[0]}_s x^{[0]} \leq b^{[0]}_s, \ x \in \mathbb{R}^{n^{[0]}} \right\} \]

We can compute layer-by-layer.
Illustrative Example

Input set:
\[ \mathcal{X}^{[0]} \triangleq \{ \mathbf{x} \mid \|\mathbf{x}\|_\infty \leq 1, \mathbf{x} \in \mathbb{R}^3 \} \]

3 inputs, 2 outputs, 7 hidden layers of 7 neurons each.

Output reachable set: union of 1250 polytopes

8000 randomly generated outputs
LEC Verification: Specification-Guided Verification for Neural Networks

Output set computation

Interval-Based Computation Procedure:

- Partition Input Space into sub-intervals
- Compute output range for sub-intervals of input
- Union of output intervals over-approximate output set

Key: How to partition the input space?
**Output set computation**

**Uniform Partition**
- Tight over-approximation *(Length of sub-interval is small)*
- Computationally expensive *(Huge number of sub-intervals)*
- Independent of specification

**Specification-Guided Partition**
- Coarse and fine partitions coexist
- Computationally efficient *(avoid unnecessary computation)*
- Non-uniform, guided by specification
LEC Verification: Specification-Guided Verification for Neural Networks

Random neural network
- Layer: 5
- Each layer: 10 neurons
- Activation function: tanh

<table>
<thead>
<tr>
<th>Method</th>
<th>Intervals</th>
<th>Verification Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>111556</td>
<td>294.37</td>
</tr>
<tr>
<td>Spec-Guided</td>
<td>4095</td>
<td>21.45</td>
</tr>
</tbody>
</table>

**Specification-guided**

- Number of partitions: 729
- Computation: ~30s

- Key results:
  - 1-2 orders of magnitude less runtime
  - 1-2 orders of magnitude less memory

**Uniform partition**

- Number of partitions: 15
- Computation: ~0.27s

**Robotic Arm Example**

- Number of partitions: 729
- Computation: ~30s

- Key results:
  - 1-2 orders of magnitude less runtime
  - 1-2 orders of magnitude less memory
LEC Model Formatting and Translation for Benchmarking

- Standard LEC representations (ONNX) & integration with standard learning frameworks
- Challenges: specification & assumption modeling, analysis parameters
Bounded Model Checking with LECs in the Loop

- Alternate iterations of reachability analysis for:
  - **nnv**: Machine learning-based / neural network controllers
  - **HyST**: Plant (and environment, noise, etc.) models

![Diagram of closed-loop system]

Iterative from time 0 to k-1

```
1: function SYSTEMREACH(W^[θ], A_i, B_i, X_0)
2:     for h = 0 : 1 : k-1 do
3:         \( G_h \leftarrow \text{networkoutput}(W^[θ], A_i, B_i, X_h) \)
4:         Compute \( X_{h+1} \) by \( X_{h+1} = \{ x | x = A_{σ(h)}x(h) + B_{σ(h)}g(x(h)), g(x(h)) \in G_h, x(h) \in X_h \} \)
5:         \( X_{[0,h+1]} = X_{h+1} \cup X_{[0,h]} \)
6:     end for
7: return \( X_k \) and \( X_{[0,k]} \)
8: end function
```
Reachability and Safety Properties

**Execution**: starting from an initial state, sequence of states visited by transitions (discrete evolution) and trajectories (continuous evolution)

**Reachable State**: state $x$ such that finite execution ends in $x$

**Set of Reachable States**: $\text{Reach}_\mathcal{A}$

**Invariant**: (safety) property $P$ that holds over all executions of $\mathcal{A}$

$\text{Reach}_\mathcal{A} \subseteq P$

$T \triangleq T_1 \lor T_2 \lor \ldots \lor T_k \lor C$

- $T$: discrete transitions
- $C$: continuous trajectories
Adaptive Cruise Control (ACC) Example

**Adaptive Cruise Control System:**
- tracks a set velocity
- maintains a safe distance

**Specification:**
\[ D_r(t) \geq \frac{D_{safe}(t)}{2} \]

where
- \( D_{safe}(t) = D_{default} + T_{gap} \times V_{ego}(t) \)
- \( D_r(t) \) is the relative distance
- \( D_{default} \) is the standstill default spacing
- \( T_{gap} \) is time gap between the vehicles
- \( V_{ego}(t) \) is velocity of the ego car
Adaptive Cruise Control (ACC) Example

- **Specification:** \( D_r(t) \geq \frac{D_{safe}(t)}{2} \), where \( D_{safe}(t) = D_{default} + T_{gap} \times V_{ego}(t) \), \( D_r(t) \) is the relative distance, \( D_{default} \) is the standstill default spacing, \( T_{gap} \) is time gap between the vehicles, \( V_{ego}(t) \) is velocity of the ego car.
ACC Closed-Loop Verification with Linear and Nonlinear Plant Models

- Plant model: 4 state variables, linear or nonlinear dynamics
- LEC: feedforward ReLU network with 5 layers and 50 neurons
- Bounded model checking: $k = 40$ steps
- Runtimes: 1-2 minutes on modern laptop, scales linearly in number of steps $k$

Nonlinear: red unsafe set and blue reachable set
ACC Closed-Loop Verification with Linear and Nonlinear Plant Models

- Plant model: 4 state variables, linear or nonlinear dynamics
- LEC: feedforward ReLU network with 5 layers and 50 neurons
- Bounded model checking: $k = 40$ steps
- Runtimes: 1-2 minutes on modern laptop, scales linearly in number of steps $k$

Nonlinear: red unsafe set and blue reachable set
# NNV-Conservativeness (CSV)

<table>
<thead>
<tr>
<th>FNN</th>
<th>Range &amp; CSV</th>
<th>Exact</th>
<th>Approximate</th>
<th>Approximate &amp; Partition</th>
<th>Mixing</th>
<th>Sherlock</th>
</tr>
</thead>
</table>
| **Abalone**  
<i> = 8,  <i> = 1,  <i> = 2,  \text{n} = 16</i> | Range | [2.18, 9.07] | [2.18, 9.07] | [2.18, 9.07] | [2.18, 9.07] | [0, 0] |
| CSV | 0% | 0% | 0% | 0% | UN |
| **Pollution**  
<i> = 24,  <i> = 3,  <i> = 3,  \text{n} = 16</i> | Range | [122.78, 206.68]  
[2.83, 13.91]  
[65.2, 116.51] | [0, 236.41]  
[0, 18.04]  
[0, 138.5] | [86.43, 222.4]  
[0, 16.13]  
[41.29, 128.22] | [122.69, 212.16]  
[2.81, 14.73]  
[65.11, 120.7] | [122.78, 206.68]  
[2.83, 13.91]  
[65.2, 116.51] |
| CSV | [0%]  
[0%]  
[0%] | [146.4%] - OA  
[37.34.9%] - OA  
[127%] - OA | [43.337%] - OA  
[25.54%] - OA  
[46.6056%] - OA | [6.53%] - OA  
[7.426%] - OA  
[8.14%] - OA | [0%]  
[0%]  
[0%] |
| **Sherlock N0**  
<i> = 2,  <i> = 1,  <i> = 1,  \text{n} = 100</i> | Range | [2.31, 8.79] | [0, 15.46] | [1.82, 9.07] | [0, 9.65] | [8.43, 10.75] |
| CSV | 0% | 102.93% - OA | 7.55% - OA | 35.63% - OA | UN |
| **Sherlock N4**  
<i> = 2,  <i> = 1,  <i> = 1,  \text{n} = 1000</i> | Range | ≈ [8.94, 128.33] | [0, 399.66] | [0, 147.19] | Timeout | [12.24, 30.62] |
| CSV | ≈ 0% | ≈ 227.27% - OA | ≈ 15.79% - OA | -- | UN |

<i> is the number of inputs, \text{o} is the number of outputs, \text{i} is the number of layers and \text{n} is the total number of neurons.  
OA: over-approximation, UN: neither an over-approximation nor an under-approximation.
## NNV-Time Reduction with Parallel Computing

<table>
<thead>
<tr>
<th>FNN</th>
<th>Cores</th>
<th>Exact</th>
<th>Approximate</th>
<th>Mixing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>T(sec)</td>
<td>R(%)</td>
<td>Output</td>
</tr>
<tr>
<td>MNIST 1</td>
<td>1</td>
<td>243.57</td>
<td>0</td>
<td>[0.91, 0.96]</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>153.33</td>
<td>37.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>142.07</td>
<td>41.67</td>
<td></td>
</tr>
<tr>
<td>(i = 784, o = 1, \ l = 6, \ n = 141)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNIST 2</td>
<td>1</td>
<td>684.6</td>
<td>0</td>
<td>[0.99, 0.993]</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>328.5</td>
<td>52.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>222.8</td>
<td>67.47</td>
<td></td>
</tr>
<tr>
<td>(i = 784, o = 1, \ l = 5, \ n = 250)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNIST 3</td>
<td>1</td>
<td>Timeout</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(i = 784, o = 1, \ l = 2, \ n = 1000)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(i\) is the number of inputs, \(o\) is the number of outputs, \(l\) is the number of layers and \(n\) is the total number of neurons. R is the reachable set computation time, R is the time reduction and Output is the output reachable set.
NNV-Verification for ACAS XU Networks

Collision avoidance using ACAS XU networks

ACAS XU Networks
- Advisory control for collision avoidance
- 45 deep neural networks
- Each network has 6 hidden layers with 50 neurons per layer (total 300 neurons)

<table>
<thead>
<tr>
<th>Property</th>
<th>FNN</th>
<th>Safety</th>
<th>Exact scheme</th>
<th>Reluplex</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RT(sec)</td>
<td>ST(sec)</td>
<td>VT(sec)</td>
<td>VT(sec)</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>$N_{2-4}$</td>
<td>safe</td>
<td>4635.7</td>
<td>2.17</td>
</tr>
<tr>
<td>$N_{2-9}$</td>
<td>safe</td>
<td>2135.5</td>
<td>2.74</td>
<td>2138.3</td>
</tr>
<tr>
<td>$N_{5-9}$</td>
<td>safe</td>
<td>1036</td>
<td>0.64</td>
<td>1036.7</td>
</tr>
<tr>
<td>$\phi_4$</td>
<td>$N_{2-9}$</td>
<td>safe</td>
<td>248.8</td>
<td>0.25</td>
</tr>
<tr>
<td>$N_{3-8}$</td>
<td>safe</td>
<td>3281.47</td>
<td>1.91</td>
<td>3283.4</td>
</tr>
<tr>
<td>$N_{5-7}$</td>
<td>safe</td>
<td>522.04</td>
<td>0.73</td>
<td>522.8</td>
</tr>
</tbody>
</table>

RT is the reach set computation time, ST is the safety checking time and VT is the total verification time.

Output reachable set for property $\phi_4$ on ACAS XU $N_{2-9}$
**Assumption:**

For any $x_1 \leq x_2$, the activation function satisfies $f(x_1) \leq f(x_2)$.

### General Neural Networks

<table>
<thead>
<tr>
<th>Name</th>
<th>Plot</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = x$</td>
</tr>
<tr>
<td>Binary step</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ 1 &amp; \text{for } x \geq 0 \end{cases}$</td>
</tr>
<tr>
<td>Logistic (a.k.a. Soft step)</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = \frac{1}{1 + e^{-x}}$</td>
</tr>
<tr>
<td>TanH</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$</td>
</tr>
<tr>
<td>ArcTan</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = \tan^{-1}(x)$</td>
</tr>
<tr>
<td>Softsign [7][8]</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = \frac{x}{1 +</td>
</tr>
<tr>
<td>Rectified linear unit (ReLU)[9]</td>
<td><img src="image" alt="Plot" /></td>
<td>$f(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ x &amp; \text{for } x \geq 0 \end{cases}$</td>
</tr>
</tbody>
</table>
Maximal Sensitivity

\[ x_0 \]

\[ \delta \]

\[ \epsilon \]

\[ \epsilon \] is called the maximal sensitivity of \( x_0 \) with respect to \( \delta \).

Input set over-approximation

Output set over-approximation
Compute Maximal Sensitivity

\[
\max_{\epsilon(x^{[\ell]}, \delta^{[\ell]})} \epsilon(x^{[0]}, \delta^{[\ell]}) = \left| f_\ell(W^{[\ell]}(x^{[\ell]} + \Delta x^{[\ell]} + \theta^{[\ell]}) - y^{[\ell]} \right|
\]

\[
y^{[\ell]} = f_\ell(W^{[\ell]}(x^{[\ell]} + \theta^{[\ell]})
\]

\[
\|\Delta x^{[\ell]}\| \leq \delta^{[\ell]}
\]
Multi-layer Neural Network

Input set

<table>
<thead>
<tr>
<th>δ</th>
<th>Num. of Simulations</th>
<th>Computational Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>2,500</td>
<td>2.29 sec</td>
</tr>
<tr>
<td>0.005</td>
<td>10,000</td>
<td>10.15 sec</td>
</tr>
<tr>
<td>0.002</td>
<td>62,500</td>
<td>55.64 sec</td>
</tr>
<tr>
<td>0.001</td>
<td>250,000</td>
<td>223.93 sec</td>
</tr>
</tbody>
</table>

Red points: 10000 random outputs. All are located in estimated reachable set.
Safe Modeling

Discretize input space by $\delta = 0.05$
Discretize input space by $\delta = 0.02$

**Computational time:**
Use cvx: $\sim 20$ min
Use linprog: $\sim 30$ sec
Pre-generate solution expression:
$\sim 0.12$ sec!
How fast?
Random 676 inputs, $\sim 0.08$ sec.

Computation cost mainly comes from the number of simulations.
Run-time Assurance (RTA) Design: Supervisory Control and Monitoring of LECs in the Loop

• Complex controller: can do anything, have LECs, etc., but only produces control inputs \((u)\) for the plant

• Check these control inputs for a finite time horizon
Closed Loop System Architecture

Environment

- Radar measurements
- Wind measurements
- Disturbances

Perception LEC
- Obstacle position and size

Motion control LEC
- Speed, direction
- Position, velocity

Ship
What is the effect of architecture on assurance of LECs?

- Decomposition may allow easier comprehension and the use of compositional techniques
- Training data required for end-to-end may be significantly higher
Verification for Machine Learning, Autonomy, and Neural Networks Survey

- “Verification for Machine Learning, Autonomy, and Neural Networks Survey”
  - Surveys most work on ML verification, including some control theory/intelligent control (guaranteeing stability while training), safe RL, and software tools
  - Weiming Xiang, Patrick Musau, Ayana Wild, Diego Manzanas Lopez, Xiaodong Yang, Joel Rosenfeld, and Taylor T. Johnson

- Draft available, open to collaborations, suggestions/missing refs, and we plan a survey/magazine paper submission, please feel free to get in touch, taylor.johnson@Vanderbilt.edu
  - https://www.overleaf.com/read/nxdtyhzhyjhz
  - QR code links to overleaf draft
# Neural Network Verification: Tools & Status

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Reference</th>
<th>Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td>[Pulina and Tacchella, 2011]</td>
<td>SMT (HySAT)</td>
</tr>
<tr>
<td>NNAF</td>
<td>[Bastani et al., 2016]</td>
<td>LP</td>
</tr>
<tr>
<td>DLV</td>
<td>[Huang et al., 2016]</td>
<td>SMT (Z3), CEGAR</td>
</tr>
<tr>
<td>Reluplex</td>
<td>[Katz et al., 2017]</td>
<td>SMT (custom), LP (GLPK)</td>
</tr>
<tr>
<td>Reverify</td>
<td>[Lomuscio and Maganti, 2017]</td>
<td>LP (Gurobi)</td>
</tr>
<tr>
<td>Planet</td>
<td>[Ehlers, 2017]</td>
<td>LP (GLPK), SAT (Minisat)</td>
</tr>
<tr>
<td>PLNN</td>
<td>[Bunel et al., 2017]</td>
<td>LP (Gurobi); Branch &amp; bound</td>
</tr>
<tr>
<td>Sherlock</td>
<td>[Dutta et al., 2017]</td>
<td>LP (Gurobi); Local search</td>
</tr>
<tr>
<td>DiffAI / AI²</td>
<td>[Gehr et al., 2018]</td>
<td>Abstract interpretation</td>
</tr>
<tr>
<td>nnv+nnmt</td>
<td>[Xiang, ..., J, 2017-2018]</td>
<td>LP (Matlab); Maximal sensitivity (non-linear activations); Reachability</td>
</tr>
</tbody>
</table>

[https://www.overleaf.com/read/nxdtyhzhypjz]  [https://arxiv.org/abs/1810.01989]
Challenges and Plans

- Alternate computations on neural network controller & plant
  - How to scale for systems where a single iteration is insufficient due to nondeterministic branching, e.g., path planning?
  - How much uncertainty to incorporate in plant & LEC analysis?
  - How to scale for deep neural networks (DNNs)?
    - State-of-the-art (all methods): $\sim 10k$ neurons, various assumptions on numbers of layers, numbers in input/output layers, etc. (See survey paper)
    - Some ideas: abstractions based on feature extraction, performing analysis in the feature space
  - Standard representations for LECs: highly recommend ONNX for NNs, need to formulate plans in the AA program for apples-to-apples comparisons of verification methods

- Other LECs / machine learning components
- Runtime monitoring, verification, and assurance
  - Environment monitoring, checking if uncertainty assumptions valid
  - Real-time computation and real-time reachability
Machine Learning, Autonomy, and Neural Network Verification Bibliographical Survey

• “Machine Learning, Autonomy, and Neural Network Verification Bibliographical Survey”
  • Surveys most work on ML verification, including some control theory/intelligent control (guaranteeing stability while training), safe RL, and software tools
  • Weiming Xiang, Joel Rosenfeld, Hoang-Dung Tran, Patrick Musau, Diego Manzanas Lopez, Ran Hao, Xiaodong Yang, and Taylor T. Johnson

• Draft available, open to collaborations, suggestions/missing refs, and we plan a survey/magazine paper submission, please feel free to get in touch, taylor.johnson@Vanderbilt.edu
  • https://www.overleaf.com/read/nxdtyhzhypjz
  • https://arxiv.org/abs/1810.01989

• QR code links to overleaf draft
Challenges and Limitations

• How do we **specify** “correctness” for machine learning components in cyber-physical systems (CPS)? Are new specification languages, such as hyperproperties and signal temporal logic (STL) expressive enough?

• What can be done for V&V of various types of machine learning components, such as **perception** versus planning/decision making/control?

• How do we analyze at **design time** such correctness?

• How do we enforce safety and other correctness criteria at **runtime** to assure autonomy?

• How can we address **scalability** for analyzing LECs, or should we consider alternative paradigms, such as **guaranteed training methods** that produce robust LECs?
Thank You
Thank You!

Questions?

• Students
  – **VU EECS**: Hoang-Dung Tran (PhD), Nate Hamilton (PhD), Ayana Wild (PhD), Patrick Musau (PhD), Xiaodong Yang (PhD), Ran Hao (PhD), Tianshu Bao (PhD), Diego Manzanas (PhD), Yuanqi Xie (PhD) Weiming Xiang (Postdoc), Joel Rosenfeld (Postdoc)
  – **UTA CSE**: Luan Viet Nguyen (PhD), Shafiul Chowdhury (PhD)
  – **UTA Alumni**: Omar Beg (PhD), Nathan Hervey (MS), Ruoshi Zhang (MS), Shweta Hardas (MS), Randy Long (MS), Rahul (MS), Amol (MS)

• Recent Collaborators
  – **UTA**: Ali Davoudi, Christoph Csallner, Matt Wright, Steve Mattingly, Colleen Casey
  – **Illinois**: Sayan Mitra, Marco Caccamo Lui Sha, Amy LaViers
  – **AFRL**: Stanley Bak and Steven Drager
  – **Toyota**: Jim Kapinski, Xiaqing Jin, Jyo Deshmukh, Ken Butts, Issac Ito
  – **Waterloo**: Sebastian Fischmeister
  – **Toronto**: Andreas Veneris
  – **ANU**: Sergiy Bogomolov
  – **UTSW**: Ian White, Victor Salinas, Rama Ranganathan

*Taylor T. Johnson*

http://www.TaylorTJohnson.com
http://www.verivital.com
Taylor.Johnson@vanderbilt.edu
Output Set of a single ReLU

Number of Cases:

\[
1 + 1 + 2^n - 2 = 2^n \\
\forall x_i > 0 \quad x_i \geq 0, x_j \leq 0
\]
Output Set of ReLU Layer

Input set of layer: $\mathcal{X}^{[\ell]} = \bigcup_{s=1}^{N_{\ell}} \mathcal{X}_s^{[\ell]}$, $\mathcal{X}_s^{[\ell]} \triangleq \{ x^{[\ell]} | A_s^{[\ell]} x^{[\ell]} \leq b_s^{[\ell]}, \ x \in \mathbb{R}^{n_s^{[\ell]} } \}$

Case 1: $\forall x_i > 0$

(a) $\begin{cases} y_1 = x_1 \quad x_1 > 0 \\ y_2 = x_2 \quad x_2 > 0 \end{cases}$

$\mathcal{X}_s^{[\ell]+} = \{ x^{[\ell]} \ | \ A_s^{[\ell]} x^{[\ell]} \leq b_s^{[\ell]} \land W_s^{[\ell]} x^{[\ell]} > -\theta_s^{[\ell]} , \ x^{[\ell]} \in \mathbb{R}^{n_s^{[\ell]} } \}$

$\forall x_i > 0$

$\mathcal{Y}_s^{[\ell]+} = \{ y \ | \ y = W_s^{[\ell]} x^{[\ell]} + \theta_s^{[\ell]} , \ x \in \mathcal{X}_s^{[\ell]+} \}$
Output Set of ReLU Layer

Input set of layer: $\mathcal{X}^{[\ell]} = \bigcup_{s=1}^{N_{\ell}} \mathcal{X}^{[\ell]}_s$, $\mathcal{X}^{[\ell]}_s \triangleq \{ x^{[\ell]} \mid A^{[\ell]}_s x^{[\ell]} \leq b^{[\ell]}_s, \ x \in \mathbb{R}^{n^{[\ell]}} \}$

Case 2: $\forall x_i \leq 0$

\[\mathcal{X}^{[\ell]}_s^- = \left\{ x^{[\ell]} \mid \begin{bmatrix} A^{[\ell]}_s \\ W^{[\ell]} \end{bmatrix} x^{[\ell]} \leq \begin{bmatrix} b^{[\ell]}_s \\ -\theta^{[\ell]} \end{bmatrix}, \ x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}\]

$\forall x_i \leq 0$

\[\mathcal{Y}^{[\ell]}_s^- = \begin{cases} \{0\}, & \mathcal{X}^{[\ell]}_s^- \neq \emptyset \\ \emptyset, & \mathcal{X}^{[\ell]}_s^- = \emptyset \end{cases}\]
Output Set of ReLU Layer

**Input set of layer:** \( \mathcal{X}[\ell] = \bigcup_{s=1}^{N_{\ell}} \mathcal{X}_s[\ell], \quad \mathcal{X}_s[\ell] \equiv \left\{ x[\ell] \mid A_s[\ell] x[\ell] \leq b_s[\ell], \ x \in \mathbb{R}^{n[\ell]} \right\} \)

**Case 3:** \( x_i \geq 0, x_j \leq 0 \)

\( \mathcal{X}^{[\ell]} = \left\{ x^{[\ell]} \mid A^{[\ell]}_{h,s} x^{[\ell]} \leq b^{[\ell]}_{h,s}, \ x^{[\ell]} \in \mathbb{R}^{n^{[\ell]}} \right\}, \)

\( A^{[\ell]}_{h,s} = \begin{bmatrix} A_s[\ell] \ 
 I - P_h W^{[\ell]} \end{bmatrix}, \quad b^{[\ell]}_{h,s} = \begin{bmatrix} b_s[\ell] \ 
 P_h \theta^{[\ell]} \end{bmatrix} \)

\( y^{[\ell]}_{h,s} = \left\{ y \mid y = P_h W^{[\ell]} x^{[\ell]} + P_h \theta^{[\ell]}, \ x \in \mathcal{X}_{h,s}^{[\ell]} \right\} \)

\( x_j \leq 0 \Rightarrow x_j = 0 \)

**Output set of layer:** \( \mathcal{Y}^{[\ell]} = \bigcup_{s=1}^{N_{\ell}} \left( \mathcal{Y}_s^{[\ell]} + \mathcal{Y}_s^{[\ell]} - \bigcup_{h=1}^{2n^{[\ell]}-2} \mathcal{Y}_{h,s}^{[\ell]} \right) \)
VeriVITAL Research on Offline Verification

- **Weiming Xiang**, Hoang-Dung Tran, **Taylor T. Johnson**, "Output Reachable Set Estimation and Verification for Multi-Layer Neural Networks", *In IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2018, March.


- **Weiming Xiang**, Hoang-Dung Tran, **Taylor T. Johnson**, "Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations", *In In Submission*, IEEE, 2018, September.

- **Weiming Xiang**, Hoang-Dung Tran, Joel Rosenfeld, **Taylor T. Johnson**, "Reachable Set Estimation and Verification for a Class of Piecewise Linear Systems with Neural Network Controllers", *In American Control Conference (ACC 2018)*, IEEE, 2018, June
VeriVITAL Research for Online Verification of LECs/LESs


• Implementations (C)
  - Cross-platform, ported/tested on: x86 and x86-64 (Windows and Linux), Arduino, ARM, MIPS
  - Java version forthcoming (~June 2018) for integration with distributed robotics control in StarL
    - [https://github.com/verivital/starl](https://github.com/verivital/starl)