Provably Correct Training of Neural Network Controllers

Yasser Shoukry
Assistant Professor
Resilient Cyber-Physical Systems Lab
Electrical Engineering and Computer Science
University of California, Irvine
Video released of Uber self-driving crash that killed woman in Arizona

New footage of the crash that killed Elaine Herzberg raises fresh questions about why the self-driving car did not stop
Video released of Uber self-driving crash that killed woman in Arizona

New footage of the crash that killed Elaine Herzberg raises fresh questions about why the self-driving car did not stop
Video released of Uber self-driving crash that killed woman in Arizona

New footage of the crash that killed Elaine Herzberg raises fresh questions about why the self-driving car did not stop

HOME FROM THE HONEYMOON, THE SELF-DRIVING CAR INDUSTRY FACES REALITY

The End of Starsky Robotics

In 2015, I got obsessed with the idea of driverless trucks and started Starsky Robotics. In 2016, we became the first street-legal vehicle to be paid to do real work without a person behind the wheel. In 2018, we became the first street-legal truck to do a fully unmanned run, albeit on a closed road. In 2019, our truck became the first fully-unmanned truck to drive on a live highway.

And in 2020, we’re shutting down.
Video released of Uber self-driving crash that killed woman in Arizona

New footage of the crash that killed Elaine Herzberg raises fresh questions about why the self-driving car did not stop

HOME FROM THE HONEYMOON, THE SELF-DRIVING CAR INDUSTRY FACES REALITY

The End of Starsky Robotics

It took me way too long to realize that VCs would rather a $1b business with a 90% margin than a $5b business with a 50% margin, even if capital requirements and growth were the same.

And growth would be the same. The biggest limiter of autonomous deployments isn’t sales, it’s safety.
Video released of Uber self-driving crash that killed woman in Arizona

New footage of the crash that killed Elaine Herzberg raises fresh questions about why the self-driving car did not stop

HOME FROM THE HONEYMOON, THE SELF-DRIVING CAR INDUSTRY FACES REALITY

The AV Space

There are too many problems with the AV industry to detail here: the professorial pace at which most teams work, the lack of tangible deployment milestones, the open secret that there isn't a robotaxi business model, etc. The biggest, however, is that supervised machine learning doesn't live up to the hype. It isn't actual artificial intelligence akin to C-3PO, it's a sophisticated pattern-matching tool.

It took me way too long to realize that VCs would rather a $1b business with a 90% margin than a $5b business with a 50% margin, even if capital requirements and growth were the same.

And growth would be the same. The biggest limiter of autonomous deployments isn’t sales, it’s safety.
Challenge: Can we systematically design “provably correct” deep neural networks?
- Theory
- Algorithms
- Implementation

It took me way too long to realize that VCs would rather a $1b business with a 90% margin than a $5b business with a 50% margin, even if capital requirements and growth were the same.

And growth would be the same. The biggest limiter of autonomous deployments isn’t sales, it’s safety.
Training Data (offline or through interaction)

NN Training

NN Weights
Imprecise Model

\[ \dot{x} = f(x, u, w) \]

System-Level Specification \( \varphi \)

Assured Architecture Synthesis


\[ \dot{x} = f(x, u, w) \]

**Imprecise Model**

**System-Level Specification**

---

**Assured Architecture Synthesis**


**Formal NN Training**


---

Training Data (offline or through interaction)
$\dot{x} = f(x, u, w)$

** Assured Architecture Synthesis


\[
\dot{x} = f(x, u, w)
\]

** Assured Architecture Synthesis

** Formal NN Training

** NN Verification

** NN Repair

- Imprecise Model
- System-Level Specification \[ \varphi \]

Training Data (offline or through interaction)

Localized Error and Plausible Fixes

- Concrete Counterexamples

- NN Weights

- Assured Architecture

- Formal NN

- Training Data

- Assured Architecture Synthesis

- Formal NN Training

- NN Verification

- NN Repair

- Formally Verified NN

---


\[
\dot{x} = f(x, u, w)
\]

** Assured Architecture Synthesis *

** Formal NN Training **

** NN Verification ***

**** NN Repair ****


\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Training Data (offline or through interaction) → Formal NN Training
\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Training Data (offline or through interaction)

Formal NN Training

\[ f(x, \mathcal{NN}(x)) \models \varphi \]
Core idea:
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Training Data (offline or through interaction)

\[ f(x, NN(x)) \models \varphi \]
Core idea:
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions
- Use reachability analysis to identify families of CPWA functions that satisfy the specs
Core idea:
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions
- Use reachability analysis to identify families of CPWA functions that satisfy the specs
Regression ReLU NN are Continuous

Core idea:
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions
- Use reachability analysis to identify families of CPWA functions that satisfy the specs
Core idea:
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[
x^{(t+1)} = f(x^{(t)}, u^{(t)})
\]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Abstract states: \( X = \{q_1, q_2, \ldots, q_n\} \)
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Abstract states: \( X = \{ q_1, q_2, \ldots, q_n \} \)

Recall:

\[ \text{NN} = \text{Continuous Piece-Wise Affine (CPWA) functions} \]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[
x(t+1) = f(x(t), u(t))
\]

Abstract states:

\[
X = \{ q_1, q_2, \ldots, q_n \}
\]

\[
u(t) = K_i x(t) + b_i
\]

Recall:

\[
\text{NN} = \text{Continuous Piece-Wise Affine (CPWA) functions}
\]
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[
x(t+1) = f(x(t), u(t))
\]

Abstract states: \( X = \{q_1, q_2, \ldots, q_n\} \)

\[
u(t) = K_i x(t) + b_i
\]

\[
P = \{(K, b) \mid K \in \mathcal{K}, b \in \mathcal{B}\}
\]

polytopic, polytopic

Recall:

\[\text{NN} = \text{Continuous Piece-Wise Affine (CPWA) functions}\]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

$$x^{(t+1)} = f(x^{(t)}, u^{(t)})$$

Abstract states: $$\mathbb{X} = \{q_1, q_2, \ldots, q_n\}$$

$$u^{(t)} = K_i x^{(t)} + b_i$$

$$\mathcal{P} = \{(K, b) \mid K \in \mathcal{K}, b \in \mathcal{B}\}$$

Controller Partitions: $$\mathbb{P} = \{P_1, P_2, \ldots, P_m\}$$
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]

**Abstract states:** \[ X = \{ q_1, q_2, \ldots, q_n \} \]

\[ u(t) = K_i x(t) + b_i \]

\[ P = \{(K, b) | K \in \mathcal{K}, b \in \mathcal{B} \} \]

**Controller Partitions:** \[ P = \{ P_1, P_2, \ldots, P_m \} \]

\[ \text{Post}(q_1, P_1) ? \]
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]

Abstract states: \[ X = \{ q_1, q_2, \ldots, q_n \} \]

\[ u(t) = K_i x(t) + b_i \]

\[ P = \{ (K, b) | K \in \mathcal{K}, b \in \mathcal{B} \} \]

Controller Partitions: \[ \mathbb{P} = \{ P_1, P_2, \ldots, P_m \} \]

Post\((q_1, P_1)\)?
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Abstract states: \( X = \{q_1, q_2, \ldots, q_n\} \)

\( u^{(t)} = K_i x^{(t)} + b_i \)

\( \mathcal{P} = \{(K, b) \mid K \in \mathcal{K}, b \in \mathcal{B}\} \)

Controller Partitions: \( \mathbb{P} = \{P_1, P_2, \ldots, P_m\} \)

Post\( (q_1, P_1) \) ?
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]

Abstract states: \( X = \{ q_1, q_2, \ldots, q_n \} \)

\[ u(t) = K_i x(t) + b_i \]

\( \mathcal{P} = \{ (K, b) \mid K \in \mathcal{K}, b \in \mathcal{B} \} \) polytopic polytopic

Controller Partitions: \( \mathcal{P} = \{ P_1, P_2, \ldots, P_m \} \)

\[ \text{Post}(q_1, P_1) = \{ q_2, q_3, q_4 \} \]
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]

Abstract states: \[ X = \{q_1, q_2, \ldots, q_n\} \]

\[ u(t) = K_i x(t) + b_i \]

\[ P = \{(K, b) \mid K \in \mathcal{K}, b \in \mathcal{B}\} \]

Controller Partitions: \[ \mathbb{P} = \{P_1, P_2, \ldots, P_m\} \]

\[ \text{Post}(q_1, P_1) = \{q_2, q_3, q_4\} \]

Transitions: \[ \text{Post}(q_i, P_j) = \{f(x, K x + b) \mid x \in q_i, (K, b) \in P_j\} \]
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]

Abstract states:
\[ X = \{q_1, q_2, \ldots, q_n\} \]

\[ u(t) = K_i x(t) + b_i \]

\[ P = \{(K, b) | K \in \mathcal{K}, b \in \mathcal{B}\} \]

polytopic \quad polytopic

Controller Partitions:
\[ \mathbb{P} = \{P_1, P_2, \ldots, P_m\} \]

\[ \text{Post}(q_1, P_1) = \{q_2, q_3, q_4\} \]

Transitions:
\[ \text{Post}(q_i, P_j) = \{f(x, Kx + b) | x \in q_i, (K, b) \in P_j\} \]

Note: Computing the \textbf{Post} operator can be done using existing techniques for reachability analysis of nonlinear systems (with the caveat that existing tools focus on partitioning the "input" space instead of the "controller" space).
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x(t+1) = f(x(t), u(t)) \]

Abstract states: \( X = \{q_1, q_2, \ldots, q_n\} \)

Controller Partitions: \( \mathbb{P} = \{P_1, P_2, \ldots, P_m\} \)

Transitions: \( \text{Post}(q_i, P_j) = \{f(x, Kx + b) \mid x \in q_i, (K, b) \in P_j\} \)
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Abstract states: \[ \mathbb{X} = \{q_1, q_2, \ldots, q_n\} \]

Controller Partitions: \[ \mathbb{P} = \{P_1, P_2, \ldots, P_m\} \]

Transitions: \[ \text{Post}(q_i, P_j) = \{f(x, Kx + b) \mid x \in q_i, (K, b) \in P_j\} \]

Specs (safety): \[ \varphi = \square \neg q_4 \]
- Use reachability analysis to identify families of CPWA functions that satisfy the specs.

\[ x(t+1) = f(x(t), u(t)) \]

Abstract states: \[ X = \{ q_1, q_2, \ldots, q_n \} \]

Controller Partitions: \[ \mathcal{P} = \{ P_1, P_2, \ldots, P_m \} \]

Transitions: \[ \text{Post}(q_i, P_j) = \{ f(x, Kx + b) | x \in q_i, (K, b) \in P_j \} \]

Specs (safety): \[ \varphi = \Box \neg q_4 \]

\[ \text{CPWA}_\varphi(q_1) = P_2 \cup P_3 \]
step 1

- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

**Abstract states:** \( X = \{q_1, q_2, \ldots, q_n\} \)

**Controller Partitions:** \( P = \{P_1, P_2, \ldots, P_m\} \)

**Transitions:** \( \text{Post}(q_i, P_j) = \{f(x, Kx + b) \mid x \in q_i, (K, b) \in P_j\} \)

**Specs (safety):** \( \varphi = \square \neg q_4 \)

\[ \text{CPWA}_\varphi(q_1) = P_2 \cup P_3 \]
\[ \text{CPWA}_\varphi(q_2) = \ldots \]
\[ \text{CPWA}_\varphi(q_3) = \ldots \]
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Abstract states: \( X = \{ q_1, q_2, \ldots, q_n \} \)

Controller Partitions: \( \mathcal{P} = \{ P_1, P_2, \ldots, P_m \} \)

Transitions: \( \text{Post}(q_i, P_j) = \{ f(x, Kx + b) \mid x \in q_i, (K, b) \in P_j \} \)

Specs (safety): \( \varphi = \square \neg q_4 \)

\[ \text{CPWA}_{\varphi}(q_1) = P_2 \cup P_3 \]
\[ \text{CPWA}_{\varphi}(q_2) = \ldots \]
\[ \text{CPWA}_{\varphi}(q_3) = \ldots \]

Note: Same can be extended to liveness properties using an abstract model built using the \textbf{Pre} operator instead of the \textbf{Post} operator
step 1
- Use reachability analysis to identify families of CPWA functions that satisfy the specs

\[ x^{(t+1)} = f(x^{(t)}, u^{(t)}) \]

Abstract states: \( X = \{q_1, q_2, \ldots, q_n\} \)

Controller Partitions: \( \mathbb{P} = \{P_1, P_2, \ldots, P_m\} \)

Transitions: \( \text{Post}(q_i, P_j) = \{f(x, Kx + b) \mid x \in q_i, (K, b) \in P_j\} \)

Specs (safety): \( \varphi = \square \neg q_4 \)

\[
\begin{align*}
\text{CPWA}_\varphi(q_1) & = P_2 \cup P_3 \\
\text{CPWA}_\varphi(q_2) & = \ldots \\
\text{CPWA}_\varphi(q_3) & = \ldots \\
\end{align*}
\]

\( f(x, K_{\text{CPWA}}(x)) \models \varphi \)

\( \forall K_{\text{CPWA}} \in \text{CPWA}_\varphi \)

Note: Same can be extended to liveness properties using an abstract model built using the \textbf{Pre} operator instead of the \textbf{Post} operator.
Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions that satisfy the specs.

**Core idea:**
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions.
- Use reachability analysis to identify families of CPWA functions that satisfy the specs.

**Step 1**

$$x^{(t+1)} = f(x^{(t)}, u^{(t)})$$

**Training Data (offline or through interaction)**

$$f(x, \mathcal{NN}(x)) \models \varphi$$

$$\forall K_{CPWA} \in CPWA_\varphi$$

**Training**

**Projection**
Core idea:
- Regression ReLU NN are Continuous Piece-Wise Affine (CPWA) functions
- Use reachability analysis to identify families of CPWA functions that satisfy the specs
Training Data (offline or through interaction) \( \xrightarrow{\text{Training}} \) \( \notin \) CPWA_\( \varphi \) \( \xrightarrow{\text{Projection}} \) \( \in \) CPWA_\( \varphi \)
Training Data (offline or through interaction) → Training → Projection

\[ \notin \text{CPWA}_\varphi \] \[ \in \text{CPWA}_\varphi \]

- For each abstract state, select one controller partition \( P^* \) from \( \text{CPWA}_\varphi \)
- For each abstract state, select one controller partition \( P^* \) from \( \text{CPWA}_\varphi \)
- For each abstract state, select one controller partition $P^*$ from $\text{CPWA}_\varphi$

- Train one “local” neural network $\text{NN}_q$ for each abstract state.
- For each abstract state, select one controller partition $P^*$ from $\text{CPWA}_q$

- Train one “local” neural network $\text{NN}_q$ for each abstract state.
- For each abstract state, select one controller partition $P^*$ from $\text{CPWA}_\varphi$.

- Train one “local” neural network $\text{NN}_q$ for each abstract state. Either using offline data (imitation learning) or interaction with the environment (Reinforcement learning).
- For each abstract state, select one controller partition $P^*$ from $\text{CPWA}_\varphi$

- Train one “local” neural network $\text{NN}_{q_i}$ for each abstract state. Either using offline data (imitation learning) or interaction with the environment (Reinforcement learning)

- Enumerate all “affine” functions $(K_i, b_i)$ in each local NN. Can be done efficiently since local NN are typically small.
- For each abstract state, select one controller partition \( P^* \) from \( \text{CPWA}_\varphi \).

- Train one “local” neural network \( \text{NN}_q \) for each abstract state. Either using offline data (imitation learning) or interaction with the environment (Reinforcement learning).

- Enumerate all “affine” functions \((K_i, b_i)\) in each local NN. Can be done efficiently since local NN are typically small.

- Projection:
\[
\min_{\widehat{W}} \| W - \widehat{W} \|
\]
\[
\text{s.t. } (K_i, b_i) \in P^* \quad \forall (K_i, b_i) \in \text{NN}_q
\]

(convex optimization problem if done layer-by-layer)
Theorem (informal):

Consider the nonlinear system $x^+ = f(x, u)$ and a safety specification $\varphi$. Define a “global” neural network controller as the composition of “local” neural network controllers:

$$\text{NN} = \text{NN}_{q_1} \| \text{NN}_{q_2} \| \ldots \text{NN}_{q_n}$$

Then:

$$f(x, \text{NN}(x)) \models \varphi$$

Theorem (informal):

Consider the nonlinear system $x^+ = f(x, u)$ and a safety specification $\varphi$. Define a “global” neural network controller as the composition of “local” neural network controllers:

$$NN = NN_{q_1} \parallel NN_{q_2} \parallel \cdots \parallel NN_{q_n}$$

Then:

$$f(x, NN(x)) \models \varphi$$

\[
\dot{\zeta}_x^{(t+\Delta t)} = \dot{\zeta}_x^{(t)} + \Delta t \, v \cos(\theta^{(t)}) \\
\dot{\zeta}_y^{(t+\Delta t)} = \dot{\zeta}_y^{(t)} + \Delta t \, v \sin(\theta^{(t)}) \\
\theta^{(t+\Delta t)} = \theta^{(t)} + \Delta t \, u^{(t)}
\]
\[ \begin{align*}
\zeta_{x(t+\Delta t)} &= \zeta_{x(t)} + \Delta t \, v \cos(\theta(t)) \\
\zeta_{y(t+\Delta t)} &= \zeta_{y(t)} + \Delta t \, v \sin(\theta(t)) \\
\theta(t+\Delta t) &= \theta(t) + \Delta t \, u^{(t)} 
\end{align*} \]

- Safe data collected and used for training
- Same data used in both experiments
\[
\zeta_x(t+\Delta t) = \zeta_x(t) + \Delta t \, \nu \cos(\theta(t))
\]
\[
\zeta_y(t+\Delta t) = \zeta_y(t) + \Delta t \, \nu \sin(\theta(t))
\]
\[
\theta(t+\Delta t) = \theta(t) + \Delta t \, \upsilon(t)
\]

- Safe data collected and used for training
- Same data used in both experiments

Formal NN Training

NN Training
\[
\begin{align*}
\zeta_{x}(t+\Delta t) &= \zeta_{x}(t) + \Delta t \cdot v \cos(\theta(t)) \\
\zeta_{y}(t+\Delta t) &= \zeta_{y}(t) + \Delta t \cdot v \sin(\theta(t)) \\
\theta(t+\Delta t) &= \theta(t) + \Delta t \cdot u(t)
\end{align*}
\]

- Safe data collected and used for training
- Same data used in both experiments

Formal NN Training

NN Training
\[
\begin{align*}
\dot{\zeta}_x(t+\Delta t) &= \dot{\zeta}_x(t) + \Delta t \ u \ \cos(\theta(t)) \\
\dot{\zeta}_y(t+\Delta t) &= \dot{\zeta}_y(t) + \Delta t \ u \ \sin(\theta(t)) \\
\theta(t+\Delta t) &= \theta(t) + \Delta t \ u(t)
\end{align*}
\]

- Safe data collected and used for training
- Same data used in both experiments

Formal NN Training

NN Training
<table>
<thead>
<tr>
<th>Workspace Index</th>
<th>Number of Abstract States</th>
<th>Number of Controller Partitions</th>
<th>Number of Safe &amp; Reachable Abstract States</th>
<th>Compute Reachable Sets [s]</th>
<th>Construct Posterior Graph [s]</th>
<th>Compute Function $P_{safe}$ [s]</th>
<th>Assign Controller Partitions [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>552</td>
<td>160</td>
<td>400</td>
<td>52.6</td>
<td>82.3</td>
<td>0.06</td>
<td>0.7</td>
</tr>
<tr>
<td>1</td>
<td>552</td>
<td>320</td>
<td>400</td>
<td>107.5</td>
<td>160.3</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>1</td>
<td>552</td>
<td>640</td>
<td>400</td>
<td>223.1</td>
<td>329.6</td>
<td>0.2</td>
<td>1.7</td>
</tr>
<tr>
<td>1</td>
<td>1104</td>
<td>160</td>
<td>800</td>
<td>108.2</td>
<td>333.0</td>
<td>0.2</td>
<td>2.3</td>
</tr>
<tr>
<td>1</td>
<td>1104</td>
<td>320</td>
<td>800</td>
<td>219.6</td>
<td>684.2</td>
<td>0.4</td>
<td>2.7</td>
</tr>
<tr>
<td>1</td>
<td>1104</td>
<td>640</td>
<td>800</td>
<td>451.5</td>
<td>1297.4</td>
<td>0.6</td>
<td>4.2</td>
</tr>
<tr>
<td>2</td>
<td>904</td>
<td>160</td>
<td>632</td>
<td>88.1</td>
<td>159.1</td>
<td>0.1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>904</td>
<td>320</td>
<td>632</td>
<td>203.6</td>
<td>313.2</td>
<td>0.2</td>
<td>1.1</td>
</tr>
<tr>
<td>2</td>
<td>904</td>
<td>640</td>
<td>632</td>
<td>393.2</td>
<td>660.8</td>
<td>0.3</td>
<td>1.7</td>
</tr>
<tr>
<td>2</td>
<td>1808</td>
<td>160</td>
<td>1264</td>
<td>202.1</td>
<td>634.6</td>
<td>0.3</td>
<td>3.4</td>
</tr>
<tr>
<td>2</td>
<td>1808</td>
<td>320</td>
<td>1264</td>
<td>388.6</td>
<td>1298.1</td>
<td>0.6</td>
<td>4.0</td>
</tr>
<tr>
<td>2</td>
<td>1808</td>
<td>640</td>
<td>1264</td>
<td>778.2</td>
<td>2564.4</td>
<td>0.9</td>
<td>5.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System Dimension $n$</th>
<th>Number of Abstract States</th>
<th>Compute Reachable Sets [s]</th>
<th>Construct Posterior Graph [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>69</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td>4</td>
<td>276</td>
<td>2.7</td>
<td>2.6</td>
</tr>
<tr>
<td>6</td>
<td>1104</td>
<td>11.7</td>
<td>34.2</td>
</tr>
<tr>
<td>8</td>
<td>4416</td>
<td>57.1</td>
<td>521.0</td>
</tr>
<tr>
<td>10</td>
<td>17664</td>
<td>258.1</td>
<td>9840.4</td>
</tr>
</tbody>
</table>
\[ \dot{x} = f(x, u, w) \]

- **Assured Architecture Synthesis**: Imprecise Model
- **Formal NN Training**: System-Level Specification
- **NN Verification**: Assured Architecture Synthesis
- **NN Repair**: Formal NN Training

**Training Data**
- (offline or through interaction)

**Localized Error and Plausible Fixes**
- Concrete Counterexamples

**References**


Synthesis of NN-based Safety Filters

Unverified Network
Synthesis of NN-based Safety Filters

Safe-by-Design Safety Filters

Unverified Network
Synthesis of NN-based Safety Filters
Synthesis of NN-based Safety Filters
Synthesis of NN-based Safety Filters

Without Root-of-Trust Network

With Root-of-Trust Network

Synthesis of NN-based Safety Filters
Imprecise Model
\[ \dot{x} = f(x, u, w) \]

System-Level Specification \( \varphi \)

---

** Assured Architecture Synthesis *

---

** Formal NN Training

---

** NN Verification

---

** NN Repair

---


Thanks!

Xiaowu Sun
Haitham Khedr
Wael Fatnassi
Ulises Santa Cruz Leal
Momina Sajid
Dr. James Ferlez

NSF
NORTHROP GRUMMAN