Towards Verification of Neural Networks for Small Unmanned Aircraft Collision Avoidance

Ahmed Irfan, Kyle D. Julian, Haoze Wu, Clark Barrett,
Mykel J. Kochenderfer, Baoluo Meng, James Lopez

Stanford University
GE Global Research

DASC 2020

Introduction

ACAS sXu

- Development led by **FAA**.
- Variant of **ACAS Xu** [1] for unmanned aircraft.
- Uses **numeric lookup tables** (large in size) for decision making.

Challenges

- **Limited memory availability** and large tables size.
- **Deep neural network** approximation of the tables **reduces** the **size** by a factor of 1000 [2].
- Big question: How can we gain **trust** in DNNs?

Our Answer

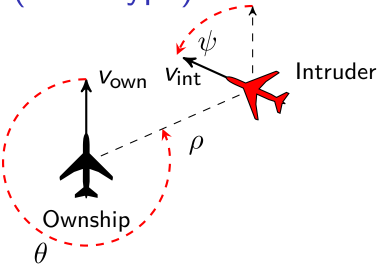
- Apply **formal verification** to gain trust in DNNs.

Agenda of the Talk

1. Background
2. DNN Training
3. Verification of DNNs
   - Local Robustness
   - Reachability Analysis
4. Conclusion

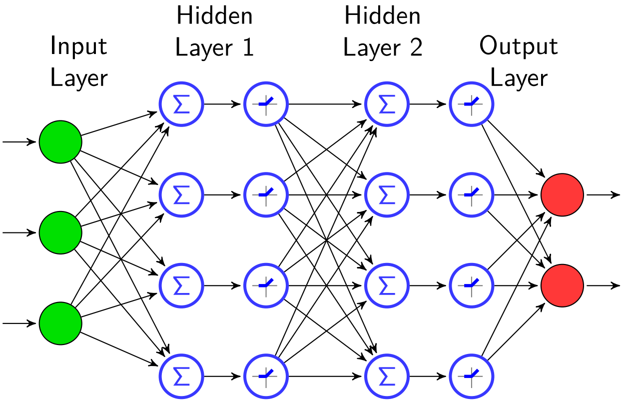
Background

ACAS sXu Table (Prototype)



Variable	Description	Values	Num
ρ (ft)	Range to intruder	$[499, 36656]$	20
θ (rad)	Bearing angle to intruder	$[-\pi, \pi]$	41
ψ (rad)	Relative heading angle of int.	$[-\pi, \pi]$	41
v_{own} (ft/s)	Ownship speed	$[100, 472]$	6
v_{int} (ft/s)	Intruder speed	$[0, 1200]$	12
τ (s)	Time to loss of vert. separation	$[0, 101]$	10
s_{adv}	Previous advisory	COC, WL, WR SL, SR	5

Deep Neural Networks (DNN)

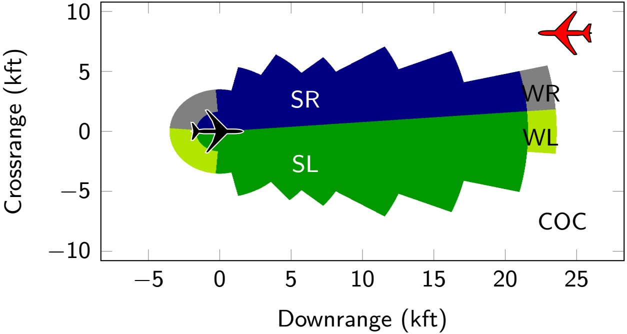


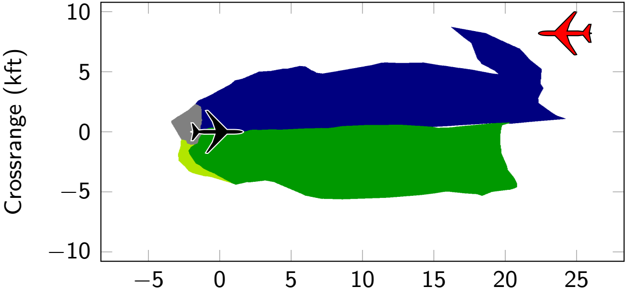
DNN Training

DNN Training

- To **reduce the time to evaluate** the networks, we trained **50 networks**: one for each combination of s_{adv} and $\tau \in \{0, 1, 5, 10, 20, 40, 60, 80, 100, 101\}$.
- **DNN architecture**: 5 inputs, 5 outputs, and 5 hidden layers.
- ρ and θ were converted to **Cartesian coordinates** x and y via $x = \rho \cos \theta$ and $y = \rho \sin \theta$.
- Each network was trained for **200 epochs** with a batch size of 512 and the **Adam gradient descent method**.
- In total, the 50 network representation requires **792 kB** of memory using 32-bit floating point precision, which is a **2600$\times$ reduction** in representation size.

Policy Comparison: Table and DNN Representation





Verification of DNNs

Verification of the DNN Representation

Verification of DNNs in isolation

- Local Robustness.

Verification of closed-loop system with DNNs

- Reachability Analysis.

Local Robustness

Intiutively

- Local robustness means that the network **behaves similar** (produces same output) on **neighboring points** to the training points.

Challenges

- Computational cost: **810,000** training points per network.
- Decision boundaries: should not expect the local robustness to hold.

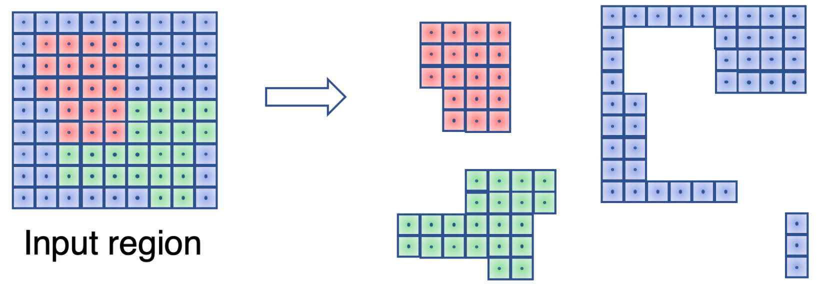
Our Approach

- **Cluster** training points into hypercubes.
- Compute **robust volume ratio** for each hypercube.

Local Robustness – A Hypercube Approach

Step-1

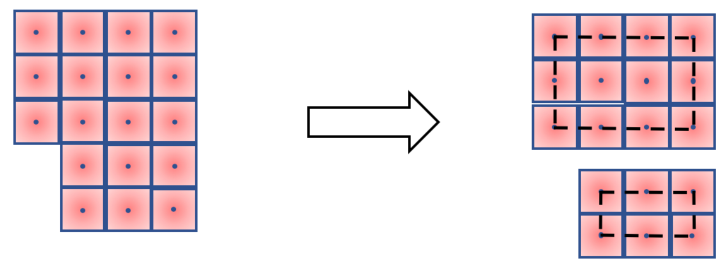
- **Decompose** the training points into clusters of **adjacent points** with the **same output label**.



Local Robustness – A Hypercube Approach

Step-2

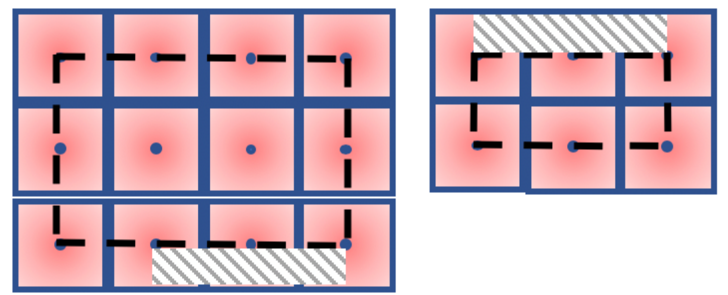
- Decompose the points in the same cluster into sets of points such that they can be **symbolically represented by a hypercube**.



Local Robustness – A Hypercube Approach

Step-3

- Compute the volume of adversarially robust regions in each hypercube.



Local Robustness – Compute robust volume ratio

Observations

- For a hypercube generated by the clustering method, it is likely that it is **not fully robust**.
- However, treating the **full volume of hypercube as unrobust is not correct**.

Details

- If a hypercube is **robust**, then we calculate its volume.
- Otherwise, if the hypercube volume is below a **certain threshold** then the hypercube is treated as **unrobust** else we partition the hypercube into k **disjoint hypercubes** and check their robustness.
- This process is continued till all the hypercubes are marked as either robust or unrobust.
- The **robust volume ratio** is computed as the ratio of the sum of volume of the robust hypercubes to the volume of all hypercubes.

Local Robustness – Results

Hypercubes Clustering Statistics

- Step-1 and Step-2 finished within **12 hours**.

	Max	Min	Median	Mean
# Clusters	7252	196	4971	5067
# Hypercubes	87631	2445	70834	75801

Local Robustness – Results

Robust Volume Computation – (Proof of concept)

- Randomly **sampled 36,375 hypercubes** (1% of the total hypercubes)
- For 36,277 hypercubes, Marabou with 4 threads completed the task within **20 minutes**.
- For 95 hypercubes, Marabou with 8 threads took less than **2 hours**.
- For the remaining 3 hypercubes, Marabou with 96 threads finished within **45 minutes**.

Robust Volume Percentage

- **Median** robust percentage: **99.66%**
- **Mean** robust percentage: **97.68%**
- 41 out of 45 networkds have robust percentage greater than 95%
- 3 networks have robust percentage above 80%
- 1 network has the percentage of 67.03%

Closed-Loop System Analysis

Observation

- DNNs are **not 100% locally robust**.
- Can we say something more about safety in the closed-loop setting?

Our approach

- Apply the reachability method proposed in [3].
- We took the dynamical model also from [3].

Closed-Loop Dynamical Model

Assumptions

- 1 ownship and 1 intruder.
- Both aircraft maintain **constant turn rates and constant speeds**.
- $v_{\text{own}} = 186\,\text{ft/s}$
- $v_{\text{int}} = 142\,\text{ft/s}$

Closed-Loop Dynamical Model

Dynamical Model

- The dynamics are a funtion of the ownship and intruder turn rates: u_{own} and u_{int} respectively.
- Advisory specifies limits on turn rates:

Aircraft	Advisory	u_{min} ($^\circ$/s)	u_{max} ($^\circ$/s)
Ownship	COC	$-\delta$	δ
Ownship	WL	$1.5 - \delta$	$1.5 + \delta$
Ownship	WR	$-1.5 - \delta$	$-1.5 + \delta$
Ownship	SR	$3.5 - \delta$	$3.5 + \delta$
Ownship	SL	$-3.5 - \delta$	$-3.5 + \delta$
Intruder	N/A	$-\delta$	δ

Closed-Loop Dynamical Model

Dynamical Model

- New positions of the ownship and the intruder:

$$
\begin{aligned}
x'_{\mathrm{own}} &= v_{\mathrm{own}} \frac{\sin(u_{\mathrm{own}})}{u_{\mathrm{own}}} \\
y'_{\mathrm{own}} &= v_{\mathrm{own}} \frac{1 - \cos(u_{\mathrm{own}})}{u_{\mathrm{own}}} \\
x'_{\mathrm{int}} &= x + v_{\mathrm{int}} \frac{\sin(\psi + u_{\mathrm{int}}) - \sin(\psi)}{u_{\mathrm{int}}} \\
y'_{\mathrm{int}} &= y + v_{\mathrm{int}} \frac{\cos(\psi) - \cos(\psi + u_{\mathrm{int}})}{u_{\mathrm{int}}} .
\end{aligned}
$$

Closed-Loop Dynamical Model

Dynamical Model

- New positions as the position of the intruder aircraft **relative to the ownship's** new position and heading direction:

$$\begin{bmatrix} x \\ y \\ \psi \\ v_{\text{own}} \\ v_{\text{int}} \\ \tau \\ s_{\text{adv}} \end{bmatrix} \leftarrow \begin{bmatrix} (x'_{\text{int}} - x'_{\text{own}}) \cos(u_{\text{own}}) + (y'_{\text{int}} - y'_{\text{own}}) \sin(u_{\text{own}}) \\ (y'_{\text{int}} - y'_{\text{own}}) \cos(u_{\text{own}}) - (x'_{\text{int}} - x'_{\text{own}}) \sin(u_{\text{own}}) \\ \psi + u_{\text{int}} - u_{\text{own}} \\ v_{\text{own}} \\ v_{\text{int}} \\ \max(0, \tau - 1) \\ s'_{\text{adv}} \end{bmatrix}$$

Reachability Analysis

Reachability Method [3]

- **Split the input region** into small cells.
- Using a DNN verification tool, **compute which advisories can be given within each cell**. (Over-approximation of the neural network.)
- Initial set of reachable $\mathcal{R}_0$ is the set of states that could occur before the neural network takes action.
- For each t, we **compute** $\mathcal{R}_{t+1}$:
  - for each cell c in $\mathcal{R}_t$, compute all the possible advisories $\mathcal{A}_c$,
  - using system dynamics to compute all the cells reachable $R_{c,a}$ in the next time step from the cell c when any advisory in $\mathcal{A}_c$ is applied,
  - $\mathcal{R}_{t+1}$ is the union of $R_{c,a}$ for every $c \in \mathcal{R}_t$ and for each advisory in $\mathcal{A}_c$.
- **Repeat** the process until an **NMAC cell is found reachable** or $\mathcal{R}$ **converges**.

Reachability Analysis – Results

Implementation and Setup Details

- Adapted the reachability code developed previously [3].
- Used **Reluval** [4] as the underlying DNN verification tool.
- Memory limit of **16GB**.

Set of Experiments

- **Precise turn rates**: $\delta = 0\,^{\circ}/\mathrm{s}$
  - **Coarse** Grid.
  - **Fine** Grid.
- Larger values of δ.
- Horizontal separation initial set.

Reachability Analysis – Results

Implementation and Setup Details

- Adapted the reachability code developed previously [3].
- Used **Reluval** [4] as the underlying DNN verification tool.
- Memory limit of **16GB**.

Set of Experiments

- **Precise turn rates**: $\delta = 0\,^{\circ}/\mathrm{s}$
  - **Coarse** Grid.
  - **Fine** Grid.
- Larger values of δ.
- Horizontal separation initial set.

Reachability Analysis – Results: Precise Turn Rates

Coarse Grid

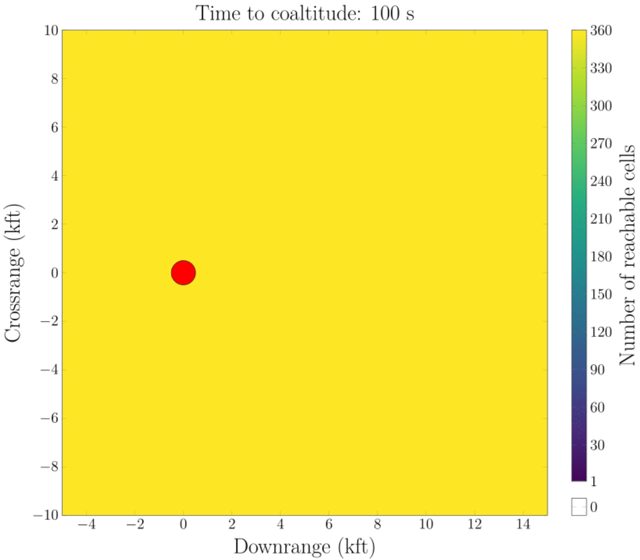
- Coarse grid discretization (**6.86 million cells**): 136 units in x and 140 units in y more dense near the NMAC region. ψ was discretized to 360 one-degree segments.
- Reluval took about **3 hours** for each network.
- Reachability analysis was **not conclusive**: NMAC was reachable in the over-approximated reachable set.

Reachability Analysis – Results: Precise Turn Rates

Fine Grid

- Fine grid discretization (**34.6 million cells**): 334 units in x and 288 units in y more dense near the NMAC region. ψ was discretized to 360 one-degree segments.
- Reluval took about **4 hours** for each network.
- Reachability analysis **concluded safe**: NMAC was not reachable in the over-approximated reachable set.

Reachability Analysis – Results: Precise Turn Rates (Fine Grid)



Points not Covered in the Talk

- Conversion from Polar to Cartesian coordinates.
- Handling of Cartesian coordinates in the computation of the robust volume ratio.
- Reachability analysis on larger values of δ and horizonal separation initial set.

Conclusion

Thank you!

Conclusion

- Presented a **methodology for formally verifying a DNN-based collision avoidance system for small unmanned aircraft**.
- Hypercube clustering can be used to verify local robustness of multiple single-points.
- DNNs are not locally robust everywhere, but using reachability analysis, we can show that the closed-loop system with the neural network cannot reach an unsafe state.

Future Work

- Improving clustering algorithm with polytopes.
- Automatic over-approximation refinement in the reachability method.
- Relaxing the assumption about constant velocities of the ownship and the intruder in the reachability analysis.

References

- L. E. Alvarez, I. Jessen, M. P. Owen, J. Silbermann, and P. Wood, "ACAS sXu: Robust decentralized detect and avoid for small unmanned aircraft systems," in *Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–9.
- K. D. Julian, M. J. Kochenderfer, and M. P. Owen, "Deep neural network compression for aircraft collision avoidance systems," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 3, pp. 598–608, 2019.
- K. D. Julian and M. J. Kochenderfer, "Guaranteeing safety for neural network-based aircraft collision avoidance systems," in *Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–10.
- S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, "Efficient formal safety analysis of neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 6369–6379.