Verifying an Aircraft Collision Avoidance Neural Network with Marabou [⋆]

Cong Liu[1], Darren Cofer[1], and Denis Osipychev[2]

[1] Collins Aerospace, {cong.liu,darren.cofer}@collins.com
[2] Boeing, denis.osipychev@boeing.com

Abstract. In this case study, we have explored the use of a neural network model checker to analyze the safety characteristics of a neural network trained using reinforcement learning to compute collision avoidance flight plans for aircraft. We analyzed specific aircraft encounter geometries (e.g. head-on, overtake) and also examined robustness of the neural network. We verified the minimum horizontal separation property by identifying conditions where the neural network can potentially cause a transition from a safe state to an unsafe state. We show how the property verification problem is mathematically transformed and encoded as linear-constraints that can be analyzed by the Marabou model checker.

1 Introduction

Machine Learning technologies such as neural networks (NN) have been used to implement advanced functionality in complex systems, including safety-critical aircraft applications. Before such systems can be deployed outside of an experimental setting, it will be necessary to show that they can meet the verification and certification requirements of the aerospace domain.

In a typical NN, much of the complexity and design information resides in its training data rather than in the actual models or code produced in the training process. One of the key principles of avionics software certification is the use of requirements-based testing along with structural coverage metrics. These activities not only demonstrate compliance with functional requirements, but are intended to expose any unintended functionality by providing a measure of completeness. However, since it is not possible to associate particular neurons or lines of code in a NN with a specific requirement, these activities cannot provide the required level of assurance [1].

Formal methods tools are being developed for NNs and may be able to address this challenge by providing a comprehensive analysis of a system over its entire input space and showing the absence of unintended behaviors. In this

[⋆] The authors wish to thank Aleksandar Zeljic for his help using Marabou. This work was funded by DARPA contract FA8750-18-C-0099. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for Public Release, Distribution Unlimited.

case study, we have used the Marabou model checker [6] to analyze a NN that was trained to compute collision avoidance flight plans for aircraft. The main contribution of the paper is to show the effectiveness of formal methods in identifying potential safety concerns in a real NN application. In fact, this NN was flight tested in a controlled experiment with two general aviation-class airplanes, but we were able to find a number of conditions which trigger unexpected (and potentially unsafe) actions [2]. Furthermore, we suggest ways in which formal analysis results can be incorporated to improve the training of future systems.

One of the unique aspects of this study is that it is focused on a NN trained using Reinforcement Learning (RL). In earlier work on the ACAS-Xu system for collision avoidance in small unmanned aircraft [4], the NN was trained using supervised learning based on a complete tabular specification of correct behavior and Reluplex [5] (a precursor of Marabou) was used to verify various safety properties. Another ACAS-Xu study [3] used formal analysis tools to show the equivalence of the NN to the tabular specification. In the current study, RL was used to compute flight plans (rather than just the avoidance maneuvers produced by ACAS-Xu), but our formal analysis exposes areas in which the training process is incomplete.

Marabou is a state-of-the-art framework for verifying deep NN. It can answer queries about NN properties by transforming the queries to a satisfiability problem. Currently it only supports linear constraints for inputs and outputs. Marabou accepts three input formats: NNet, TensorFlow and ONNX. In the case study, we exported the NN model parameters from the learning environment and encoded them in the NNet format. We used the Marabou Python interface to encode the constraints and perform the verification.

The collision avoidance NN and the Marabou verification scripts are available at https://github.com/darrencofer/NFM-2023-case-study.

2 Aircraft Collision Avoidance Neural Network

We study the automated aircraft collision avoidance system developed described in [2]. The system's core is a NN model pre-trained on a surrogate simulation using RL. The NN model modified the course of the controlled airplane (ownship) to provide a safe distance to another aircraft (intruder) and return to the original course when safe. The RL environment simulates various potential collision scenarios with aircraft performance similar to a Cessna 208 Caravan. The 2-D position range is [-10000, 10000] m $\times$ [-10000, 10000] m. The heading range is [-180, 180] degree. The aircraft speed range is [50, 70] mps. The required minimum separation distance (MSD) is 2000 m. The initial and goal position are randomly generated and remain fixed during each training scenario. During the encounter, while the intruder maintains a constant direction and speed, the ownship adjusts the flight direction and speed. Not maintaining the MSD results in a penalty, while returning to the original route results in a reward.

In the experiments, a number of policies were developed. We chose a NN that only controls the flight direction (i.e. constant speed). It consists of 8 input nodes,

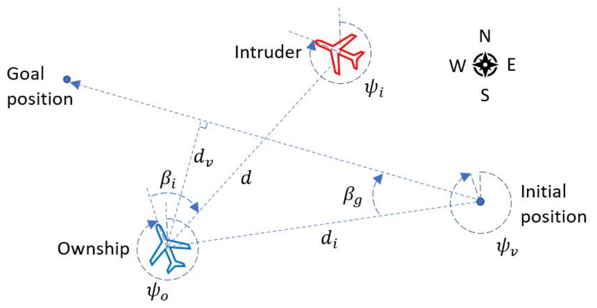

Fig. 1: System geometry

1 output node, and 2 hidden layers with 256 nodes each. All hidden nodes use rectified linear unit (ReLU) as the activation function. The output node uses Tanh as the activation function. Marabou does not support Tanh activation functions, so for the verification we removed the Tanh activation function and mapped its outputs (e.g. value or range) back to the corresponding function input.

The NN inputs are: $\{d, d_v, d_i, v_r, \beta_i, \psi_r, \beta_g, \beta_v\}$, where d is distance from ownship to intruder, d_v is distance to vector, d_i is distance to initial position, v_r is relative speed, β_i is angle to intruder, ψ_r is relative heading, β_g is angle to goal, β_v is angle to vector. The vector is from the initial position to the goal. We define $v_r = v_i/v_o - 1$, $\psi_r = \psi_i - \psi_o$, $\beta_v = \psi_v - \psi_o$, where v_i is intruder speed, v_i is onwship speed, ψ_i is intruder heading, ψ_o is ownship heading, ψ_v is vector heading. The system geometry in shown in Figure 1.

Note that $d_v = d_i \sin \beta_g$. This means that the NN inputs are not completely independent. We capture this dependence by encoding the relation as a constraint. Since Marabou only supports linear constraints, we set β_g as a constant in each analysis.

All NN inputs are normalized: $d, d_v, d_i \in [0, 1]$, $v_r \in [-0.3, 0.4]$, $\beta_i, \psi_r, \beta_g, \beta_v \in [-1, 1]$. The NN output range is (-1, 1) due to the Tanh function. It is linearly mapped to (-3, 3) to compute the turn rate (ω), unit in degree per second. A positive and negative value indicates turning right and left, respectively.

3 Verifying Minimum Separation Distance

The reachability analysis of a closed-loop neural network controller is known to be challenging. Instead, we examine the condition where the ownship transitions from a safe state ($d = MSD$) to a unsafe state ($d < MSD$). This indicates that the distance function is decreasing at the MSD boundary. We mathematically derive the derivative of the distance function and check when the neural network will generate an output action that causes the derivative to be negative at the boundary. Although the derivative function itself is non-linear (e.g. involving

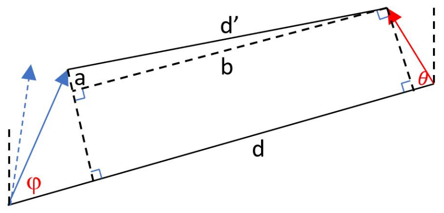

Fig. 2: Distance between ownship and intruder changes during Δt

trigonometric functions), we found that the safety conditions can be characterized by a set of linear constraints, which can be handled by Marabou. Figure 2 illustrates the derivative calculation. The blue and red arrows represents the ownship and the intruder movement during interval Δt, respectively, with d and d' being the distance at time t and $t + \Delta t$.

The distance d' satisfies $d'^2 = a^2 + b^2$, where:

$a = v_o \Delta t \sin\varphi - v_i \Delta t \sin\theta$,

$b = d - v_o \Delta t \cos\varphi - v_i \Delta t \cos\theta$

and v_o, v_i denotes the speed of the ownship and the intruder, respectively. Ignoring the higher order terms, we have $(d'^2 - d^2)/\Delta t = -2d(v_o \cos\varphi + v_i \cos\theta)$. Applying the chain rule $\Delta d^2/\Delta t = 2d\Delta d/\Delta t$, we have $\Delta d/\Delta t = -(v_o \cos\varphi + v_i \cos\theta)$. By definition: $\varphi = \beta_i - \omega\Delta t$, $\theta = 180 - \beta_i + \psi_r$. Letting $\alpha = \beta_i - \psi_r$, we rewrite the derivative as $\dot{d} = v_i \cos\alpha - v_o \cos\varphi$.

Assuming $v_i = v_o$, $\dot{d} < 0$ implies $\cos\alpha < \cos\varphi$. If $0 \leq \alpha \leq 180, 0 \leq \varphi \leq 180$, then it implies $\alpha > \varphi$ (i.e. $\omega\Delta t > \psi_r$). This means that at the MSD boundary, the neural network has to generate a turn angle that is less than the relative heading to prevent the distance from decreasing. Note that the turn rate ω is limited to the range $(-3, 3)$ degree per second and $\Delta t = 1$ second in simulation. This means if $\psi_r \leq -3$ or $\psi_r \geq 3$, no matter what the neural network output is, the derivative will always be negative or positive, respectively. Thus, in our analysis we restrict to the scenarios where $\psi_r \in (-3, 3)$. In other words, we only look for the scenarios where the MSD violation could be avoided, but the neural network does not generate such output.

Results For the analysis, we sampled the intruder angle between 0 and 180 degrees, and found MSD violations for each intruder angle. The simulation used in the RL training process makes it unlikely that the critical conditions where safety is violated (e.g. MSD boundary, relative heading range) are are reached very often, meaning that the NN likely has insufficient training in this region to make safe decisions.

4 Robustness Analysis

Robustness analysis helps us to understand the stability of the NN controller. In most cases, the output produced by the NN should not change dramatically in reponse to small input perturbations (such as sensor noise). We can perform a δ-local-robustness [5] to quantify the bounded-input/bounded-output stability of the NN.

To perform this analysis, we generated five arbitrary points covering a range of input conditions and NN outputs. We computed a constant δ for an input point x such that for all inputs $x' : \|x - x'\|_\infty \leq \delta$, the neural network output will not change sign (e.g., changing from turning left to right).

Table 1: Robustness analysis results.

	$\delta = 0.1$	$\delta = 0.05$	$\delta = 0.02$	$\delta = 0.01$
Point 1 (weak right turn)	SAT	SAT	SAT	UNSAT
Point 2 (strong left turn)	SAT	SAT	SAT	UNSAT
Point 3 (strong right turn)	SAT	SAT	UNSAT	UNSAT
Point 4 (strong right turn)	SAT	UNSAT	UNSAT	UNSAT
Point 5 (strong left turn)	SAT	SAT	UNSAT	UNSAT

Results The robustness analysis results are summarized in Table 1. SAT results mean an adversarial input was found, while UNSAT results mean no such inputs exist. The results show that the neural network may be not robust. In particular at Point 2, a small input perturbation ($\delta = 0.02$) causes the ownship to change from turning strong-left to right. This may lead to unstable behavior in which the aircraft oscillates between left and right turns.

5 Specific Scenarios

We examine six encounter scenarios, where there are expected aircraft maneuvers (e.g. staying on course vs. turning left or right). We check whether the action generated by the neural network aligns with expectations. The following scenarios were considered.

Head-on The ownship is on course and both airplanes are about to have a head-on collision. We expect the ownship shall make a turn to avoid collision.

Overtake The ownship is on course while the intruder flies in the same direction and approaches from behind. We expect the ownship shall turn to avoid collision.

Parallel same direction The ownship is on course while the intruder flies side by side in the same direction and is close. We expect the ownship shall not fly towards the intruder.

Parallel opposite direction The ownship is on course while the intruder flies side by side in the opposite direction and is *dangerously* close. We expect the ownship shall not fly towards the intruder.

Approach from right The ownship is on course while the intruder approaches the ownship from right and is *dangerously* close. We expect the ownship shall turn left.

Far away The ownship is on course while the intruder is far away. We expect the ownship shall stay on its course and will not make large turns.

Table 2: Verification of mid-air encounter scenarios. All times are in seconds.

Scenario	Constraints	Result	Time
Head-on	$d = 2000$, $d_i \geq 0$, $0.4 \geq v_r \geq -0.3$, $d_v = 0$, $\beta_i \geq 0$, $\psi_r = 180$, $\beta_g = 0$, $\beta_v = 0$, $out = 0$	SAT	0.02
Overtake	$d \leq 2500$, $d_i \geq 0$, $0.4 \geq v_r \geq -0.3$, $d_v = 0$, $\beta_i \geq 0$, $\psi_r = 0$, $\beta_g = 0$, $\beta_v = 0$, $out = 0$	SAT	4.0
Parallel same direction	$d \leq 2500$, $d_i \geq 0$, $0.4 \geq v_r \geq -0.3$, $d_v = 0$, $\beta_i \geq 0$, $\psi_r = 0$, $\beta_g = 0$, $\beta_v = 0$, $out \geq 0$	UNSAT	12.3
Parallel opposite direction	$d = 2000$, $d_i \geq 0$, $0.4 \geq v_r \geq -0.3$, $d_v = 0$, $90 \geq \beta_i \geq 0$, $\psi_r = 180$, $\beta_g = 0$, $\beta_v = 0$, $out \geq 0$	SAT	0.03
Approach from right	$d = 2000$, $d_i \geq 0$, $0.4 \geq v_r \geq -0.3$, $d_v = 0$, $\beta_i \geq 0$, $\psi_r = -90$, $\beta_g = 0$, $\beta_v = 0$, $out \geq 0.1$	SAT	0.08
Far away	$d = 10000$, $d_i \geq 0$, $0.4 \geq v_r \geq -0.3$, $d_v = 0$, $\beta_g = 0$, $\beta_v = 0$, $out \leq -5$	SAT	18.7

Results The verification results of the six scenarios are summarized in Table 2. The NN generated an outputs violating expectations in all but one of the six scenarios. As in the MSD analysis, we believe that the RL training method did not provide sufficient training data to cover these critical scenarios. All experiments were performed on a Linux server with Intel Xeon E5-2698 v4 CPU @ 2.20 GHz and about 504 GB memory.

6 Conclusion and Future Work

We analyzed an aircraft collision avoidance NN using Marabou. We verified the minimum horizontal separation property, analyzed robustness of the NN, and investigated specific interesting scenarios. The results suggest that the RL NN training approach was insufficient to guarantee safety of the system in many critical scenarios. This shows the value of formal analysis for identifying unintended behaviors the may be present in a NN.

The counterexamples generated in the verification of a property could be used to better train the NN. The counterexamples often represent hard-to-reach corner cases. We could directly use them to train the NN in a Supervised Learning environment, because usually there are well-defined expected NN outputs. Or we could adjust the RL setup by directly setting these scenarios as the initial states.

It would be interesting to combine the forward reachability analysis of NN with our MSD property verification. So that a simulation trace from the initial state to the violation state is generated. Also recall that at certain conditions, due to the turn rate limit, the MSD property violation is unavoidable. We could use backward reachability analysis to compute the corresponding previous system states and actions, until the NN could potentially generate an action to deviate from the collision course. These system states and desired actions could also be added to the training set.

References

1. Cofer, D.: Unintended behavior in learning-enabled systems: Detecting the unknown unknowns. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC). pp. 1–7 (2021)
2. Cofer, D., Sattigeri, R., Amundson, I., Babar, J., Hasan, S., Smith, E.W., Nukala, K., Osipychev, D., Moser, M.A., Paunicka, J.L., Margineantu, D.D., Timmerman, L., Stringfield, J.Q.: Flight test of a collision avoidance neural network with run-time assurance. In: 2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC). pp. 1–10 (2022)
3. Damour, M., De Grancey, F., Gabreau, C., Gauffriau, A., Ginestet, J.B., Hervieu, A., Huraux, T., Pagetti, C., Ponsolle, L., Clavière, A.: Towards certification of a reduced footprint acas-xu system: A hybrid ml-based solution. In: Habli, I., Sujan, M., Bitsch, F. (eds.) Computer Safety, Reliability, and Security. pp. 34–48. Springer International Publishing, Cham (2021)
4. Irfan, A., Julian, K.D., Wu, H., Barrett, C., Kochenderfer, M.J., Meng, B., Lopez, J.: Towards verification of neural networks for small unmanned aircraft collision avoidance. In: 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC). pp. 1–10. IEEE (2020)
5. Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J.: Reluplex: An efficient smt solver for verifying deep neural networks. In: Majumdar, R., Kunčak, V. (eds.) Computer Aided Verification. pp. 97–117. Springer International Publishing, Cham (2017)
6. Katz, G., Huang, D.A., Ibeling, D., Julian, K.D., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D.L., Kochenderfer, M.J., Barrett, C.W.: The marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification. p. 443–452 (2019)