# Flight Test of a Collision Avoidance Neural Network with Run-Time Assurance

Darren Cofer[1], Ramachandra Sattigeri[1], Isaac Amundson[1], Junaid Babar[1], Saqib Hasan[1]
Eric Smith[2], Karthik Nukala[2],
Denis Osipychev[3], Lucca Timmerman[3], Dragos Margineantu[3], James Paunicka[3], Matthew Moser[3]
[1] Collins Aerospace, Minneapolis MN
[2] Kestrel Institute, Palo Alto CA
[3]Boeing, Everett WA

Aircraft systems have requirements for airworthiness certification that present barriers to the use of machine learning technologies such as neural networks. Showing that a component or system is correct and does no harm through behaviors that were unintended by designers or unexpected by operators is a critical aspect of the certification process. In a typical machine learning application, much of the complexity and design information resides in its training data rather than in the actual models or software produced. This means that it is generally not possible to determine the correctness of a neural network by examining its implementation or tracing specific design elements back to requirements.

Our team is developing assurance technologies that can support the use of machine learning in the design of safety-critical aircraft systems. These capabilities have been integrated on Boeing's autonomy demonstrator aircraft to show that they can provide evidence of correct operation and safety guarantees needed by real aircraft. In previous work we have used a run-time assurance architecture (RTA) to ensure the safety of an autonomous neural network-based aircraft taxiing application. In our current work we have applied run-time assurance along with formal methods modeling and analysis tools to an airborne collision avoidance system based on a neural network. This system was demonstrated in-flight and shown to correctly monitor neural network operation and intervene when needed to prevent violation of the "stay well clear" safety requirement relative to an intruder aircraft. Thirty-six test conditions were flown with various collision course geometries, with relative heading angles of 45, 90, 135, 180 (head-on), 225, 270, and 315 degrees leading to a pending collision, to test the robustness of the neural net trajectory generator and the RTA mitigations.

The RTA architecture includes a run-time monitor that provides an independent assessment of the avoidance flight plans generated by the neural network and a safe (but less optimal) backup planner. The results of the assessment are evaluated by a decision logic component which selects (based on a tabular specification of safety rules) a flight plan that will ensure safe flight. The core decision logic code is synthesized from a formal specification, with most of the synthesis steps producing machine-checked proofs of their correctness. The RTA architecture has been modeled in the Architecture Analysis and Design Language (AADL) and formally analyzed to show that it maintains the system safety requirements.

Autonomous stay-well-clear and collision avoidance capabilities are critical to safe flight of military and commercial uncrewed aircraft. These autonomous capabilities can also be valuable for enhancing the safety of flight for piloted aircraft by providing pilot cueing. Run-time assurance approaches, with run-time monitoring of machine learning software functions integrated with contingency management functionality, will enable safe use of neural networks in bringing breakthrough autonomy capabilities to aircraft.

In this paper we will discuss:

- The assurance challenges for the use of neural networks in safety-critical aircraft applications
- The autonomy framework and aircraft used to demonstrate the collision avoidance neural network capability
- The run-time assurance architecture developed to guarantee the absence of unintended behavior resulting from the neural network
- The formal methods assurance technologies applied within the architecture, including analysis of the architecture design and synthesis of decision logic from a formal specification
- Flight testing, results obtained for various test scenarios, and lessons learned from the demonstration