Assured Collision Avoidance for Learned Controllers: A Case Study of ACAS Xu

Gokul Puthumanaillam*, Manav Vora†, Taha Shafa‡, Yangge Li§, Sayan Mitra¶, and Melkior Ornik‖
University of Illinois Urbana-Champaign, Urbana, USA

This paper introduces a novel approach to verification of neural network controlled systems, combining the capabilities of the nnenum framework with the Verse toolkit. Addressing a critical gap in the traditional verification process which often does not include the system dynamics analysis while computing the neural network outputs, our integrated methodology enhances the precision and safety of decision-making in complex dynamical systems. By iteratively verifying neural network decisions and propagating system states, we maintain an accurate representation of the system's behavior over time, a vital aspect in ensuring operational safety.

Our approach is exemplified through the verification of the neural network controlled Airborne Collision Avoidance System for Unmanned Aircraft (ACAS Xu). We demonstrate that the integration of nnenum and Verse not only accurately computes reachable sets for the UAS but also effectively handles the inherent complexity and nonlinearity of the system. The resulting analysis provides a nuanced understanding of the system's behavior under varying operational conditions and interactions with other agents, such as intruder aircraft. The comprehensive simulations conducted as part of this study reveal the robustness of our approach, validating its effectiveness in verifying the safety and reliability of learned controllers. Furthermore, the scalability and adaptability of our methodology suggest its broader applicability in various autonomous systems requiring rigorous safety verification.

I. Nomenclature

W_i	=	weight matrix for layer i of neural network
b_i	=	bias vector for layer i of neural network
s_k	=	state vector at time instant k
S_0	=	set of initial states
(x_k, y_k)	=	position coordinates of UAS in x-y plane at time instant k
θ_k	=	heading of the UAS with respect to x-axis at time instant k
ρ	=	distance between ego aircraft and intruder aircraft
ϕ	=	angle to intruder with respect to ego heading
ψ	=	heading of the intruder with respect to ego
Z	=	ego's sensor that gives the position of the intruder
D	=	set of decisions generated by ACAS Xu
$D_{verified}$	=	subset of decisions after formal verification of D using nnenum
S_t	=	one-step reachable set of states

*Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. Email: gokulp2@illinois.edu

†Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. Email: mkvora2@illinois.edu

‡Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. Email: tahaas2@illinois.edu

§Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. Email: li213@illinois.edu

¶Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. Email: mitras@illinois.edu

‖Department of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, USA. Email: mornik@illinois.edu

II. Introduction

As aerospace systems increasingly move towards automation, there arises the need for efficient approximations and compact representations of complex control strategies. Such algorithms often involve the use of neural networks to compress data, presenting a promising solution in environments with limited processing capabilities. One such method is the Airborne Collision Avoidance System (ACAS) Xu [1]. However, ensuring the safety and reliability of these approximations remains a pressing challenge, especially in the context of implementing these methods in safety-critical systems like aircraft. Current verification methods [2–5] face significant limitations in handling the complexity and nonlinearity of these systems, necessitating novel approaches to guarantee their safety and reliability.

Given this context, our work puts forward an innovative approach for verification of *Neural Network Controlled Systems (NNCS)*, utilizing Verse, a versatile library for scenario creation, simulation, and verification [6], in conjunction with nnenum, a specialized neural network verification tool [7, 8]. Our methodology capitalizes on nnenum's adeptness at handling various activation functions and its efficient use of abstraction refinement techniques. This enables the precise verification of decisions emanating from the ACAS Xu neural networks, thereby giving a subset of feasible decisions at each time step. We supplement this capability with Verse's robust hybrid system verification techniques which are capable of handling multiple agents and uncertain dynamics. The reachability analysis of Verse, facilitated by the DryVR [9] engine, allows for efficient exploration of all possible system states, enabling a meticulous evaluation of system behavior and evolution over time thereby acting as a system verification tool.

Various methods of hybrid system verification have been implemented in the past for safety-critical control [10, 11]. Earlier methods focus on verification using CTL specification, bisimulation, and quotient transition systems for discrete model approximations [12–14]. In more recent years, the verification of hybrid systems has shifted away from accurately modeling the system behavior to using reachable sets [15–17] to calculate provably safe trajectories. Recent published methods [18, 19] are effective verification tools using model checking for reachable set computation, but lack the scalability and real-time computation potential of Verse using nnenum-based neural network verifiers.

We integrate Verse and nnenum based neural network verifiers with ACAS Xu system to generate safey assertions for different scenarios by performing verification of the ACAS Xu NNCS while also considering the dynamics. This study contributes to the growing discourse on the safety and reliability of neural networks in the aviation industry, laying the groundwork for future research and development.

III. Technical Background

The foundational components of the proposed approach involves three main components: the ACAS Xu system, the Verse library, and the nnenum-based neural network verifiers.

A. Airborne Collision Avoidance System (ACAS) Xu

The Airborne Collision Avoidance System X (ACAS X) [20, 21] is a family of mid-air collision avoidance systems, with ACAS Xu being an extended variant designed specifically for unmanned operations. The decision-making logic of ACAS Xu is grounded in a large lookup table, consisting of all possible state-action pairs, which is approximately of 2 gigabytes in size. It is developed using frameworks of dynamic programming and Markov decision processes. Table 1 lists the available advisories for ACAS Xu. These advisories are the possible actions that an aircraft can take while being driven by the ACAS Xu NNCS. Previous work [22] has proposed a compressed neural network representation of the lookup table in an effort to improve the storage efficiency and to enable computationally efficient decision-making. Such a system makes use of neural networks to inform its decisions, a setup known as a neural network control system (NNCS). Contrary to conventional machine learning systems, the ACAS Xu system operates and makes decisions based on the aircraft dynamics and a compressed policy within the neural networks, without engaging in learning from data.

Action	Description
Clear-of-conflict (COC)	No maneuvering necessary
Weak Left (WL)	Weak left turn at 1.5 deg/s
Weak Right (WR)	Weak right turn at 1.5 deg/s
Strong Left (SL)	Strong left turn at 3 deg/s
Strong Right (SR)	Strong right turn at 3 deg/s

Table 1 Turn advisories

B. nnenum neural network verifiers

The nnenum [8] neural network verifier forms the second main component of the technical foundation. The verifier is designed to analyze neural networks. A critical downside of traditional neural networks is their sensitivity to minute targeted changes in the inputs. This sensitivity makes these networks susceptible to adversarial attacks which can manipulate the network's decision-making. This threat is particularly significant in safety-critical and mission-critical systems, necessitating robust verification methods. nnenum tool responds to this need by developing algorithms to reason formally over the function computed by a neural network, thereby guaranteeing safety.

nnenum focuses on densely connected [23], feedforward neural architectures employing *ReLU activation function*, which is defined as $\mathrm{ReLU}(x) = \max(0, x)$. These networks are tasked with mapping inputs to outputs by executing a series of computations determined by their architecture. A typical neural model is constructed with l layers, where each layer i is characterized by a weight matrix W_i and a bias vector b_i. For a given input $x_0 \in \mathbb{R}^n$, the output $y_k \in \mathbb{R}^m$ generated by such a neural construct is computed as follows:

$$
\begin{aligned}
z^{(1)} &= W_1 x_0 + b_1, \quad y_1 = \phi(z^{(1)}) \\
z^{(2)} &= W_2 y_1 + b_2, \quad y_2 = \phi(z^{(2)}) \\
&\vdots \\
z^{(l)} &= W_l y_{l-1} + b_l, \quad y_k = \phi(z^{(l)})
\end{aligned}
$$

Here, $z^{(i)}$ and y_i denote the pre-activation and post-activation values at layer i, respectively. The function ϕ represents the activation function applied element-wise.

Definition 1 (Output Range) *[8] Given a neural network that computes the function NN and an input set $I \subseteq \mathbb{R}^{n_i}$, the output range is the set of possible outputs of the network, when executed from a point inside the input set, defined as $Range(NN, I) = \{y_k \mid y_k = NN(y_1), y_1 \in I\}$.*

Definition 2 (Verification Problem for Neural Networks) *[8] Given a neural network that computes the function NN, an input set $I \subseteq \mathbb{R}^{n_i}$, and an unsafe set $U \subseteq \mathbb{R}^{n_o}$, the verification problem for neural networks is to check if $Range(NN, I) \cap U = \emptyset$.*

In order to solve the neural network verification problem, nnenum relies on *abstraction refinement*, a technique which uses abstractions to compute over-approximations of the set of possible outputs of a neural network. If the abstract system fails to verify the property, refinement is performed to achieve a finer abstraction, proceeding to exact analysis if necessary. This process is particularly useful when the abstractions can prove there is no intersection with unsafe states, potentially increasing the speed of the verification process. nnenum is built on a unique approach of using fast abstractions [24] for speed, combined with refinement through Rectified Linear Unit (ReLU) splitting [25] to increase accuracy.

C. Verse

We use nnenum to verify the neural network output and generate a subset of all possible decisions. However, in the context of system verification, nnenum does not take the dynamics into account. Hence we nnenum's neural network verification with Verse's system verification capabilities.

Definition 3 (System Verification) *System verification refers to the process of assessing whether a dynamical system, characterized by its state evolution over time under given inputs, adheres to specified safety and operational criteria. Unlike neural network verification, which focuses on the correctness of output for given inputs in isolation, system verification incorporates the dynamics of the entire system. This includes continuous control aspects (like flight dynamics in aircraft systems) and discrete decision-making components (such as advisories from neural networks). The goal is to determine whether, for a given set of initial states and inputs, the system can evolve into states that violate safety or operational constraints. This involves the computation of reachable sets, which represent all possible states that the system can enter over time.*

Verse [6] is a powerful open-source library that broadens the applications of hybrid system verification technologies to scenarios involving multiple interacting decision-making agents. Hybrid systems, characterized by both continuous and discrete dynamics, are often complex and challenging to analyze, a notable example being the ACAS Xu system. It involves continuous control (the flight dynamics of the aircraft) and discrete decision-making (advisories from neural networks). Verse allows the ability to create decision-making agents, operating on specified decision logics, in Python and offers functions for performing systematic simulation and verification through reachability analysis. The computation of reachable sets is a fundamental problem in the analysis of dynamical systems. It allows for the prediction of the system's future states based on its current state and input, enabling a thorough evaluation of the system's behavior and the identification of potential safety violations. To represent the reachable set, Verse constructs a *decision tree*, where each branch represents a different possible state of the system. To make the verification process efficient, Verse performs this analysis incrementally: instead of starting from scratch each time, it saves and reuses data from previous runs, thereby avoiding redundant computations while constructing the decision tree.

Verse relies on a reachability engine—a tool which performs the computations for reachability analysis. Verse currently implements algorithms from [9] for performing reachability analysis for the continuous part of hybrid systems. The algorithm is particularly efficient at calculating how small changes in the input can affect the output of the system, a property known as *sensitivity*. Verse computes sensitivity of continuous trajectories by learning a discrepancy function from simulation data with probably approximately correct (PAC) guarantees. Additionally, Verse is also capable of handling uncertain dynamics within the system. This means that even when the system's behavior is defined by complex differential equations and is subject to noise or uncertainity in transition, Verse can still calculate the set of all possible states the system can reach using a decomposition function.

Lastly, one of Verse's unique capabilities is its *map abstraction*, where a map defines a set of paths that agents can follow. Despite the potentially infinite number of paths, these paths fall into a finite set of categories, termed *track modes*. This abstraction allows for the transferability of an agent's decision logic across different maps with the same track modes, making Verse a versatile solution for diverse multi-agent scenarios. This feature is particularly beneficial in the contexts of motion control, air traffic management, and tactical collision avoidance scenarios such as ACAS Xu.

IV. Problem Formulation

We address a scenario involving two Unmanned Aircraft Systems (UAS): an *ego* aircraft and an *intruder* aircraft. The primary focus is on the ego aircraft, which is equipped with the Airborne Collision Avoidance System for Unmanned aircraft (ACAS Xu). This system outputs a compressed neural network that forms the crux of the ego UAS's decision-making logic. The objective of the ego UAS is to perform collision avoidance maneuvers. The intruder aircraft operates under a predetermined decision logic. It is programmed to perceive its environment as devoid of any potential threats, thereby consistently opting for a *Clear of Conflict* (COC) action (Table 1. To comprehensively address this scenario, we will first take a look at the UAS dynamics and the ACAS Xu NNCS.

A. UAS Dynamics

We model the UAS as a Dubins aircraft. At any discrete time instant k, the state of the UAS is defined by the state vector s_k, which encompasses

$$s_k = (x_k, y_k, \theta_k, k),$$

where x_k and y_k represent the UAS's position in the horizontal plane, and θ_k is its heading angle. The index k denotes the discrete time step, capturing the temporal aspect of the UAS's trajectory.

The motion of the UAS is governed by the following set of difference equations:

$$
\begin{aligned}
x_{k+1} - x_k &= v \cdot \cos(\theta_k), \\
y_{k+1} - y_k &= v \cdot \sin(\theta_k), \\
\theta_{k+1} - \theta_k &= u,
\end{aligned}
$$

where v is the constant forward velocity, and u represents the control action which is obtained from the decision logic.

B. Controller

The ACAS Xu neural network was trained using polar coordinate inputs. Hence, the state of the UAS needs to be converted into the input space of the neural network. The ACAS Xu neural network takes as input the distance (ρ), angle to intruder wrt ego heading (ϕ) and heading of the intruder with respect to ego (ψ). The conversion between the Dubin's model parameters and the NNCS inputs is as follows:

$$
\begin{aligned}
\rho &= \sqrt{(x_{int} - x_{ego})^2 + (y_{int} - y_{ego})^2}, \\
\phi &= atan2\left(\frac{y_{int} - y_{ego}}{x_{int} - x_{ego} + \rho}\right) - \psi_{ego}, \\
\psi &= \psi_{int} - \psi_{ego}.
\end{aligned}
$$

C. Problem Statement

In this work, we consider the problem of verification of the decision outputs provided by the ego UAS's compressed neural network and using them to perform reachability analysis and safety assertions for the system dynamics. This verification process is critical for two main reasons:

1) **Safety Violations:** We aim to systematically evaluate whether the decisions proposed by the neural network uphold the stringent safety standards necessary in aviation, particularly in scenarios where the intruder UAS's behavior is non-responsive or predictable in its persistence to maintain course.
2) **Reachable Sets Analysis:** Another crucial aspect of our study is to analyze the *reachable sets*, which represent the set of all possible positions the ego UAS can occupy over time, given its range of initial state and potential maneuver decisions. This analysis is pivotal in understanding the limits of the ego UAS's maneuvering capabilities under various conditions and decision logic outcomes.

V. Proposed Method

While existing methodologies [26, 27] provide frameworks for verifying neural network-controlled systems, they typically do not account for the impact of control decisions on the system's dynamics. Our methodology enhances the verification process by not only examining the decisions themselves but also evaluating the influence these decisions have on the system's dynamic behavior. Prior to this work, the Verse tool did not natively support the verification of decisions derived from neural network controllers. Subsection V.A discusses the methodology for decision generation by ACAS Xu and the verification of the neural network using nnenum. Subsection V.B discusses the integration of nnenum with Verse for state propagation and temporal analysis in hybrid systems.

A. Decision Generation

Let us denote the set of initial states of the ACAS Xu system as S_0. Each element $s_{0,i} \in S_0$, where $i \in \{1, \ldots, n\}$, is an initial state that encapsulates the aircraft's initial position and velocity. All these elements of the initial state s_i cumulatively contribute to the decision-making process of the system.

Given these initial states, the ACAS Xu system generates a set of decisions, represented as D. The decisions are the outcomes of the function f_{NN}, which assigns appropriate aircraft action to each possible state after operating on the aircraft dynamics and the compressed policy within the neural network. Thus, $D = f_{NN}(S_0)$.

The function f_{NN}, despite being capable of approximating from data, is susceptible to being sensitive to minor alterations in the inputs. This sensitivity is especially prominent in the context of decision-making networks, where it might lead the system to fail due to sensor noise. To analyse the robustness of the system in presence of uncertainty, we

employ a *targeted sensor noise* approach. In the specific scenario of our ego airplane, full observability of the intruder's state is attained through a sensor Z. The sensor gives the position of the intruder (pos) at every time instant.

Taking into consideration the measurement noise, the measurement equation for Z is shown below,

$$f(\hat{pos}) = f(pos) + \epsilon, \tag{1}$$

where ϵ is the noise component of signal measured by the ego. $\hat{pos}$ follows a Gaussian distribution with arbitrarily chosen mean and a variance.

In the context of the ACAS Xu system, we subject the function computed by the neural network, f_{NN}, to formal reasoning with the aim of proving properties over the inputs and outputs of the network. We perform this reasoning by running the network inputs through nnenum, which evaluates the possible decision set D of the neural network. The neural network is parsed layer by layer, with each neuron within a layer being examined individually. Based on the neuron's input, the set can be split into two using an intersection operation, a process referred to as refinement. This refinement process is recursive, continuing until the property under investigation is either proven or found to be unsafe, or there are no remaining neurons where overapproximation was performed. The outcome of this process is a set of verified decisions,

$$D_{verified} = f_{nnenum}(D), D_{verified} \subseteq D. \tag{2}$$

This increases the dependability of the decisions generated by the ACAS Xu system.

The nnenum tool, on its own, is primarily focused on verifying the outputs of neural networks and does not directly handle the complexities and dynamics of hybrid systems. These systems require a broader analysis, which not only considers the validity of neural network outputs, but also includes the exploration of system states resulting from these decisions under uncertain dynamics. Although nnenum provides a rigorous check on the reliability of decisions generated by the neural network, it stops short of evaluating the potential system states that can result from these decisions. These limitations underline the need for extrapolating this method that extends beyond the scope of existing tools. By integrating the decision verification capabilities of nnenum with the reachability analysis of Verse, we propose a comprehensive verification methodology that addresses these challenges.

B. Decision Propagation and Verification

Verse plays a critical role in propagating the set of current states and the corresponding set of verified decisions to the subsequent time step. This propagation function computes the the one-step reachable set of states S_{t+1} at the subsequent time step based on the current states S_t and the verified decisions $D_{verified}$.

$$S_{t+1} = f_{Verse}(S_t, D_{verified}) \tag{3}$$

where f_{Verse} is the function in Verse that propagates the dynamics. It calculates This process accurately carries the uncertainty associated with the decisions derived from the ACAS Xu system through the dynamics of the aircraft, resulting in a new reachable set of states at the next time step.

The final step of our methodology involves the iterative application of this process across multiple time steps. This iterative mechanism, which effectively extends the state propagation and decision verification processes over time, allows for the progressive exploration of the system's behavior.

Each cycle of this iterative process generates a subsequent reachable set, representing the ensemble of all possible states that the system can achieve at a particular time step. By conducting this process repeatedly over a defined period, we can accumulate a sequence of reachable sets, each associated with a distinct time step:

$$S_{t+n} = f_{Verse}(S_{t+n-1}, f_{nnenum}(f_{NN}(S_{t+n-1}))), \tag{4}$$

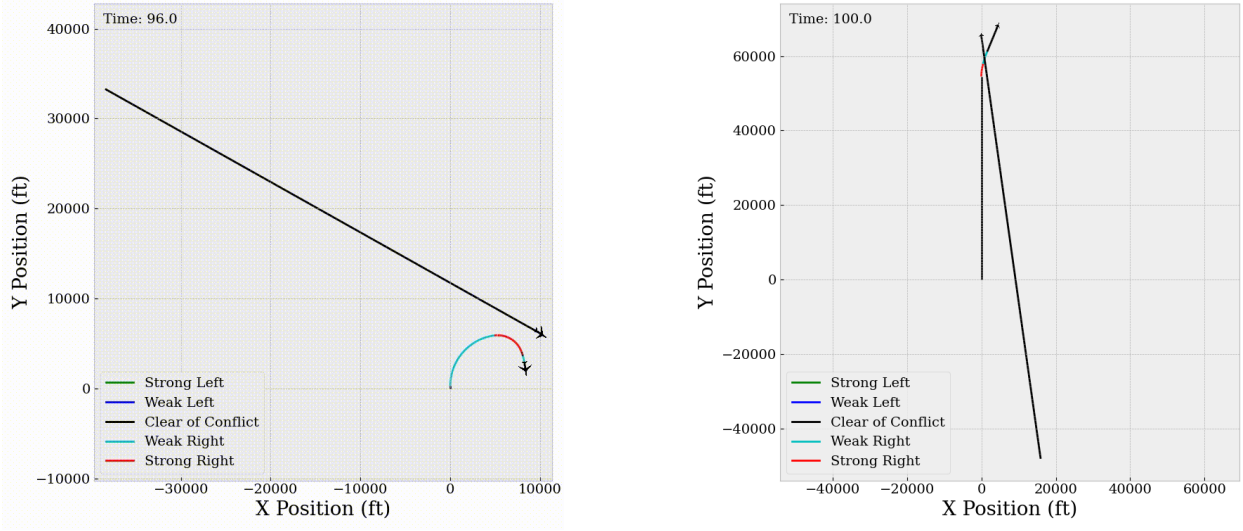
where S_{t+n} represents the reachable set at the $(t+n)$-th time step. The function f_{Verse}, takes as inputs the set of states from the previous time step, S_{t+n-1}, and the set of decisions verified by nnenum, which is derived from the neural network's response to the states at the previous time step $f_{nnenum}(f_{NN}(S_{t+n-1}))$. Equation (4) illustrates the recursive nature of the process. Doing so offers a cohesive, iterative approach, ensuring that decisions made by a neural network control system can be systematically evaluated, verified, and propagated through time. We will now discuss the integration methodology in the next subsection.

C. Integration Approach

The integration algorithm of Verse with nnenum, initiates with two aircraft (ego and intruder), each characterized by distinct state functions and a set of control actions. The algorithm employs a queue-based approach, initializing with the respective initial states and control actions of the aircraft. It iteratively processes elements from the queue, conducting reachability analysis using Verse for the current state and control actions (equation (3)), and subsequently performs a safety check on the reach set. Following this, the algorithm generates potential decisions from the ACAS Xu system and verifies these using nnenum (equation (2)). For each combination of verified control actions, the algorithm updates the queue with the new states and incremented time, ensuring a comprehensive exploration of all possible state-action pairs over the simulation duration (equation (4)).

VI. Analysis

In our analysis, we explore two distinct scenarios facilitated by varying the ACAS Xu random seeds. These seeds generate random initial states for the intruder aircraft, as well as distinct velocity profiles for both the intruder and the ego aircraft (as illustrated in Figure 1). Each scenario presents a unique set of initial conditions, thereby providing a comprehensive overview of the system's behavior under varied operational contexts. To ensure a rigorous system verification, we systematically extend our analysis across a broad spectrum of initial states for the ego aircraft.



(a) (b)

Fig. 1 ACAS Xu neural network closed loop simulation for cases 1 and 2. It shows the trajectories of the ego and intruder aircrafts.

A. Scenario 1:

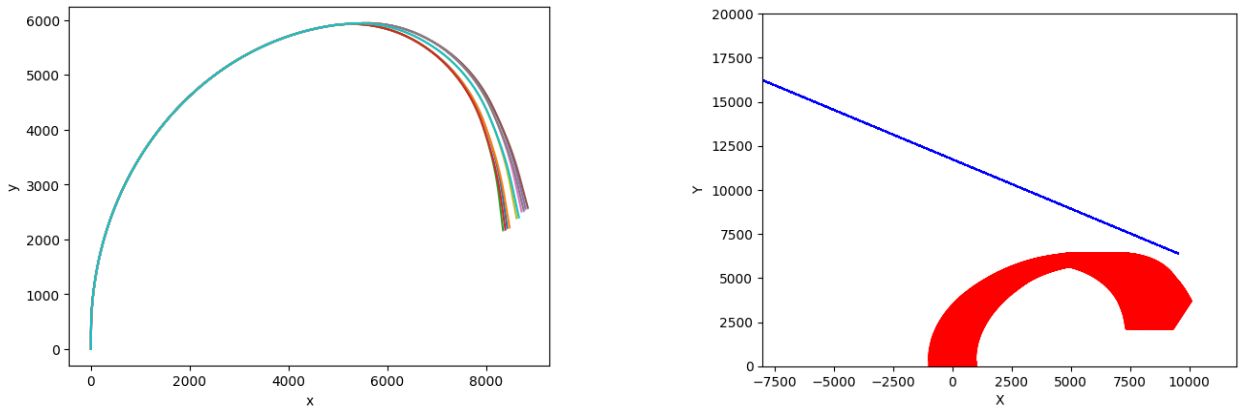
Let us consider a particular ACAS Xu scenario involving the ego and intruder aircrafts as shown in Figure 1a. As described in Section IV, the intruder always takes the COC decision (Table 1). The ego aircraft performs a right turn maneuver, consisting of COC, WR and SR, which is generated using ACAS Xu's NNCS. We follow the steps outlined in subsection V.C to integrate nnenum with Verse to perform verification of the ACAS Xu system. The verification is performed for a set of initial starting positions of the ego aircraft given by

$$s_0 = [x, 0, \pi/2, 0] \; \forall \, x \in [-1000, 1000].$$

The other parameters used for simulation are given in Table 2.

Parameter	Value
Intruder initial state	[-38430.17, 33296.38, 5.77, 0.0]
Ego velocity	148 ft/s
Intruder velocity	579 ft/s
Simulation time	96 s
ACAS Xu time step	1 s

Table 2 Parameters for scenario



(a) (b)

Fig. 2 Comparative analysis of trajectory generation and verification for the scenario depicted in Figure 1a. (a) shows a subset of trajectories generated using nnenum ($D_{verified}$), and (b) presents the verification results of a multi-aircraft scenario, involving ego and intruder aircrafts, using the integrated tool.

Figure 2a illustrates the subset of potential trajectories derived from the decision-making process as determined by nnenum. In contrast, Figure 2b provides a visual representation of the corresponding reach tubes, showcasing the trajectories' spatial evolution over time taking into account the dynamics and sensor noise. These reach tubes are obtained by augmenting nnenum's capabilities through integration with the Verse toolkit, enabling a more comprehensive analysis of the system's dynamic behavior under the specified decision set. As can be seen from Figure 2b, none of the reach tubes intersect with the intruder's path. Hence, for this scenario, the decisions taken by ACAS Xu can be asserted to be safe.

B. Scenario 2:

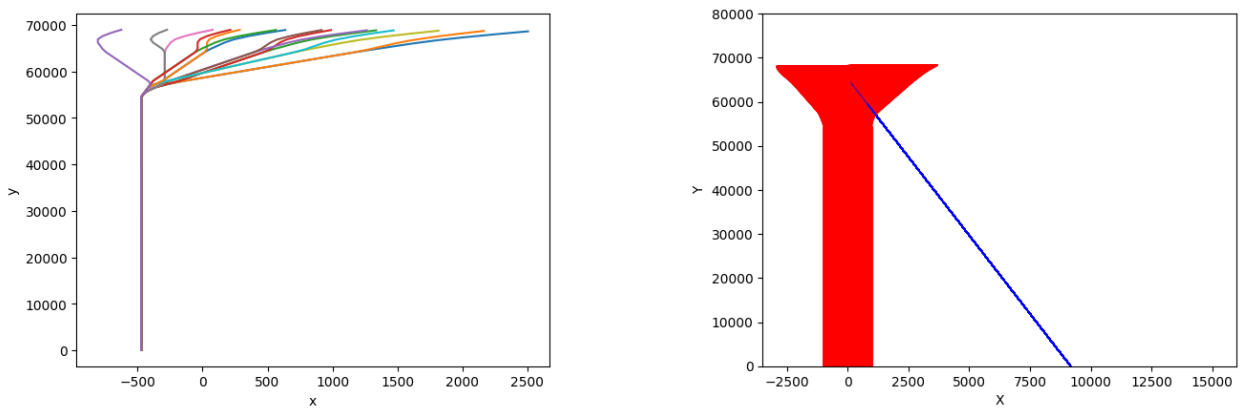
We consider the scenario shown in Figure 1b. The initial position of the ego aircraft remains the same as that of scenario 1. i.e.

$$s_0 = [x, 0, \pi/2, 0] \; \forall \; x \in [-1000, 1000].$$

The other parameters used for simulation are given in Table 3.

Parameter	Value
Intruder initial state	[15929.77, -48053.48, 1.71, 0.0]
Ego velocity	690 ft/s
Intruder velocity	1147 ft/s
Simulation time	100 s
ACAS Xu time step	1 s

Table 3 Parameters for verification



(a) (b)

Fig. 3 Comparative analysis of trajectory generation and verification for the scenario depicted in Figure 1b. (a) shows a subset of trajectories generated using nnenum ($D_{verified}$), and (b) presents the verification results of a multi-aircraft scenario, involving ego and intruder aircrafts, using the integrated tool.

In Figure 3a, we observe the range of potential trajectories computed by the nnenum framework, each corresponding to different decisions made within the neural network control logic. Figure 3b complements this by showcasing the reach tubes that account for the system's dynamic responses to these decisions. A critical observation from Figure 3b is the intersection of approximately half of the reach tubes with the intruder's predicted trajectory, suggesting a potential for unsafe encounters in those cases. Nonetheless, our methodology's consideration of system dynamics reveals that there exists a subset of initial conditions and associated decision paths where safety is maintained—indicated by the absence of overlap between the reach tubes and the intruder's trajectory. This nuanced analysis enables us to identify specific operational states wherein the ACAS Xu system's advisories lead to safe outcomes, despite the presence of risk in other scenarios.

The comprehensive simulation of both the ego and intruder aircraft using Verse enables a more detailed and accurate verification of the ACAS Xu system compared to just using nnenum. Such a holistic approach is imperative for ensuring robustness and reliability in real-world operational environments where unpredictability is a common challenge.

VII. Conclusions and Future Work

This paper presents a novel approach to the verification of neural network-controlled systems, achieved through the integration of the nnenum framework with the Verse toolkit. Our method addresses a significant challenge in the field of dynamical systems, particularly in the verification of neural network outputs and their interaction with system dynamics.

Traditionally, the verification of neural network outputs has been conducted independently from the analysis of the corresponding system dynamics, leading to potential inaccuracies and unsafe decisions, as highlighted by Bak et al. [28]. Our integrated approach overcomes this issue by coupling the verification of neural network outputs with the temporal propagation of system states. By iteratively verifying decisions and propagating states, our method maintains an accurate and dynamic representation of the system, thus enhancing the precision of the verification process.

The proposed methodology significantly contributes to formal verification in dynamical systems. It introduces a robust means of handling the complexity and nonlinearity inherent in many dynamical systems and their learned controllers. By accurately computing reachable sets for systems operating based on learned controllers, our approach provides a substantial verification advantage over verifiers that do not take the dynamics into account.

The use of Verse alongside nnenum ensures safety within a broad spectrum of operational conditions and against a range of uncertainties. Our methodology's efficacy is demonstrated through rigorous simulations, including a case-study of the neural network-controlled ACAS Xu system. These simulations, which serve as illustrative examples, incorporate varying initial conditions and the presence of intruder aircraft. They have validated the effectiveness of our integrated system in ensuring safety and reliability. This work contributes to the existing body of knowledge by providing a scalable and adaptable methodology that can extend to other autonomous systems requiring reliable safety verification, from air traffic management to autonomous vehicular systems in urban environments.

Looking ahead, our future work aims to address current limitations by exploring more sophisticated models that

account for additional real-world factors such as dynamic environmental conditions and variable performance parameters. We also intend to enhance the computational efficiency of the system to enable real-time verification, crucial for practical deployment.

Acknowledgments

This work was supported by NASA under grants 80NSSC22M0070 and 80NSSC21K1030, and by the Air Force Office of Scientific Research under grant FA9550-23-1-0131. The simulations and experiments made use of computing resources provided by University of Illinois Urbana-Champaign.

References

[1] Owen, M. P., Panken, A., Moss, R., Alvarez, L., and Leeper, C., "ACAS Xu: Integrated collision avoidance and detect and avoid capability for UAS," *38th AIAA/IEEE Digital Avionics Systems Conference*, 2019, pp. 1–10.

[2] Jia, K., and Rinard, M., "Verifying Low-Dimensional Input Neural Networks via Input Quantization," *Static Analysis: 28th International Symposium*, 2021, p. 206–214.

[3] Ivanov, R., Weimer, J., Alur, R., Pappas, G. J., and Lee, I., "Verisig: Verifying safety properties of hybrid systems with neural network controllers," *22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, p. 169–178.

[4] Haluszczynski, A., and Räth, C., "Controlling nonlinear dynamical systems into arbitrary states using machine learning," *Scientific Reports*, Vol. 11, No. 1, 2021.

[5] Jafarpour, S., Harapanahalli, A., and Coogan, S., "Interval reachability of nonlinear dynamical systems with neural network controllers," *Learning for Dynamics and Control Conference*, 2023, pp. 1–14.

[6] Li, Y., Zhu, H., Braught, K., Shen, K., and Mitra, S., "Verse: A python library for reasoning about multi-agent hybrid system scenarios," *Computer Aided Verification*, 2023, pp. 351–364.

[7] Bak, S., "Nnenum: Verification of ReLU neural networks with optimized abstraction refinement," *13th NASA Formal Methods Symposium*, 2021, p. 19–36.

[8] Bak, S., Tran, H.-D., Hobbs, K., and Johnson, T. T., "Improved geometric path enumeration for verifying ReLU neural networks," *32nd International Conference on Computer Aided Verification*, 2020, pp. 66–96.

[9] Fan, C., Qi, B., Mitra, S., and Viswanathan, M., "DryVR: Data-driven verification and compositional reasoning for automotive systems," *Conference on Computer Aided Verification*, 2017, pp. 441–461.

[10] Schilling, C., Forets, M., and Guadalupe, S., "Verification of neural-network control systems by integrating Taylor models and zonotopes," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 2022, pp. 8169–8177.

[11] Julian, K. D., and Kochenderfer, M. J., "A reachability method for verifying dynamical systems with deep neural network controllers," , 2019.

[12] Chutinan, A., "Hybrid system verification using discrete model approximations," Ph.D. thesis, Carnegie Mellon University, 1999.

[13] Chutinan, A., and Krogh, B. H., "Approximating quotient transition systems for hybrid systems," *2000 American Control Conference*, 2000, pp. 1689–1693.

[14] Pappas, G. J., "Bisimilar linear systems," *Automatica*, Vol. 39, No. 12, 2003, pp. 2035–2047.

[15] Stursberg, O., and Krogh, B. H., "Efficient representation and computation of reachable sets for hybrid systems," *6th International Workshop on Hybrid Systems: Computation and Control*, 2003, pp. 482–497.

[16] Schupp, S., "State set representations and their usage in the reachability analysis of hybrid systems," Ph.D. thesis, RWTH Aachen University, 2019.

[17] Kochdumper, N., and Althoff, M., "Reachability analysis for hybrid systems with nonlinear guard sets," *23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–10.

[18] Bak, S., and Chaki, S., "Verifying cyber-physical systems by combining software model checking with hybrid systems reachability," *13th International Conference on Embedded Software*, 2016, pp. 1–10.

[19] Xiang, W., Tran, H.-D., and Johnson, T. T., "Reachable set computation and safety verification for neural networks with ReLu activations," *arXiv preprint arXiv:1712.08163*, 2017.

[20] Kochenderfer, M. J., *Decision Making under Uncertainty: Theory and Application*, 2015, pp. 249–273.

[21] Kochenderfer, M. J., Holland, J. E., and Chryssanthacopoulos, J. P., "Next generation airborne collision avoidance system," *Lincoln Laboratory Journal*, Vol. 19, No. 1, 2012, pp. 17–33.

[22] Julian, K., Kochenderfer, M., and Owen, M., "Deep neural network compression for aircraft collision avoidance systems," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2019, pp. 598–608.

[23] Huang, G., Liu, Z., Maaten, L. V. D., and Weinberger, K. Q., "Densely connected convolutional networks," *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.

[24] Shankar, N., *Verification by Abstraction*, Springer, 2003.

[25] Rössig, A., and Petkovic, M., "Advances in verification of ReLU neural networks," *Journal of Global Optimization*, Vol. 81, 2021, pp. 109–152.

[26] Clavière, A., Asselin, E., Garion, C., and Pagetti, C., "Safety verification of neural network controlled systems," *51st Annual IFIP/IEEE International Conference on Dependable Systems and Networks Workshops*, 2021, pp. 47–54.

[27] Zhao, Q., Chen, X., Zhao, Z., Zhang, Y., Tang, E., and Li, X., "Verifying neural network controlled systems using neural networks," *25th ACM International Conference on Hybrid Systems: Computation and Control*, 2022.

[28] Bak, S., and Tran, H.-D., "Neural network compression of ACAS Xu early prototype is unsafe: closed-loop verification through quantized state backreachability," *14th NASA Formal Methods Symposium*, 2022, pp. 280–298.