

Scalable Surrogate Verification of Image-based Neural Network Control Systems Using Composition and Unrolling

Feiyang Cai^{1,3}, Chuchu Fan², Stanley Bak¹

¹Department of Computer Science, Stony Brook University

²Department of Aeronautics and Astronautics, Massachusetts Institute of Technology

³School of Computing, Clemson University

feiyang@clemson.edu, chuchu@mit.edu, stanley.bak@stonybrook.edu

Abstract

Verifying safety of neural network control systems that use images as input is a difficult problem because, from a given system state, there is no known way to mathematically model what images are possible in the real-world. We build upon recent work that considers a surrogate verification approach, training a conditional generative adversarial network (cGAN) as an image generator in place of the real world. This setup enables set-based formal analysis of the closed-loop system, providing analysis beyond simulation and testing. While existing work is effective on small examples, excessive overapproximation both within a single control period (one-step error) and across multiple periods (multi-step error) limits its scalability. We propose approaches to overcome these errors. First, we address one-step error by composing the system’s dynamics along with the cGAN and neural network controller, without losing the dependencies between input states and the control outputs as in the monotonic analysis of the system dynamics. Second, we reduce multi-step error by repeating the single-step composition, essentially unrolling multiple steps of the control loop into a large neural network. We then leverage existing network verification algorithms to compute accurate reachable sets for multiple steps, avoiding the accumulation of abstraction error at each step. We demonstrate the effectiveness of our approach in terms of both accuracy and scalability using two case studies. On the aircraft taxiing system, the converged reachable set is 175% larger using the prior baseline method compared with our proposed approach. On the emergency braking system, with 24× the number of image output variables from the cGAN, the baseline method fails to prove any states are safe, whereas our improvements enable set-based safety analysis.

Introduction

Neural networks are key enablers of image-based control systems, where applications span from autonomous vehicles (Chen et al. 2015) to industrial robotics (Levine et al. 2018). Unfortunately, neural networks rarely come with guarantees of robustness or worst-case behaviors due to the inherent non-zero error rates and are often susceptible to adversarial attacks (Boloor et al. 2020; Cai, Li, and Koutsoukos 2020). In safety-critical scenarios, ensuring the safety and reliability of neural network control sys-

tems (NNCS) is paramount. The deployment of such systems demands rigorous *closed-loop* verification to show that hazardous situations are avoided. Although considerable strides have been made in verifying NNCS (Sun, Khedr, and Shoukry 2019; Schilling, Forets, and Guadalupe 2022), verification of *image-based* NNCS has recently gained attention (Hsieh et al. 2022; Astorga et al. 2023; Păsăreanu et al. 2023; ArjomandBigdeli, Mata, and Bak 2024). This task is more difficult—image observations are high-dimensional inputs and vision networks are typically more complex than control networks, employing convolutional or transformer layers. Scalability aside, such verification problems are even hard to formulate, since reasoning over the set of images possible in the real world is not a precise mathematical statement. Recent approaches have used geometric camera models to accurately represent the perception process, but they are not applicable to complex scenarios (Habeeb et al. 2023).

One approach to this problem is using generative models to replace the perception system for the verification of image-based NNCS (Katz et al. 2022). This first trains a conditional generative adversarial network (cGAN) (Mirza and Osindero 2014) to approximate the perception system, generating images based on system states. The concatenation of cGAN and controller results in a unified network controller with low-dimensional state inputs. This surrogate controller allows existing verification methods (Xiang and Johnson 2018), which combine neural network verification tools and reachability analysis, to be applied.

However, the previous verification methodology (Katz et al. 2022) suffers from being overly conservative, computing the reachable sets with significant overapproximation, caused by two main sources: *one-step error* and *multi-step error*. Within a single control period, the prior work computes the intervals of the control outputs and then applies monotonic analysis of the system dynamics to obtain the reachable sets, ignoring the dependencies between the input states and control outputs, causing one-step error. Additionally, discrete abstraction of reachable sets after each step introduces multi-step error. The interaction of these two sources of error produces considerable overapproximation, potentially leading to false positives. As image-based NNCS grow in complexity and incorporate modern neural network architectures, extending the verification techniques to encompass these complexities presents additional challenges.

This work builds upon the surrogate verification approach using cGAN and focuses on mitigating overapproximation within the existing method. The first improvement is to compose the discrete-time dynamics along with the cGAN and neural network controller. This composition preserves the state-control dependencies, reducing one-step error. We introduce two distinct composition options to accommodate varying network scales and system dynamics. One option is incorporating the dynamics as an additional layer within the networks, allowing direct handling by neural network verification tools. This approach may encounter challenges when nonlinear dynamics are not supported by these tools or when overapproximation of nonlinear functions in the dynamics causes infinite outputs. The other option combines neural network exact analysis (Bak 2021) with the reachability algorithms from hybrid systems (Althoff, Stursberg, and Buss 2008). This method computes more accurate reachable sets using the star set representation (Duggirala and Viswanathan 2016), further reducing the one-step error. This approach is applicable when the network's scale is moderate and the network exclusively employs ReLU activations. Instead of limiting the verification scope to a single step, we propose another improvement by unrolling multiple steps of the control loop into a unified, larger neural network. We then leverage state-of-the-art network verification tools, specifically $\alpha\text{-}\beta$ -CROWN (Zhang et al. 2018) and nnenum (Bak 2021), to compute reachable sets over multiple steps. This unrolling strategy reduces the frequency of abstraction operations and, consequently, minimizes the multi-step error.

We evaluate our method with two case studies. The first focuses on an autonomous aircraft taxiing system (Katz et al. 2022), featuring a feed-forward neural network controller. The evaluation demonstrates that the proposed approach significantly reduces overapproximation, and the reachable set upon convergence using the prior baseline method is 175% larger than with our proposed approach. Therefore, our method can verify scenarios where the existing approach fails. The second case is an advanced emergency braking system (Cai and Koutsoukos 2020), incorporating modern image-based controllers in the loop. This work stands as the first attempt to conduct reachability analysis for a closed-loop system with convolutional and transformer layers.

Background

This section begins by presenting the system model and formulating the verification problem for image-based NNCS. We then introduce the surrogate approximation of the perception system using generative models. Finally, we discuss the challenges of verifying such surrogate systems.

Problem Formulation

Definition 1 (System Model) Consider an autonomous system with an image-based neural network controller, as depicted in Fig. 1. At step k , the ego system is in state $x_k \in \mathcal{X}$. The perception system P , employs a camera sensor to observe the surrounding environment $e_k \in \mathcal{E}$, and translates these observations into a representative image $o_k \in \mathcal{O}$. The image o_k is then input into an image-based

controller C to generate a control command $u_k \in \mathcal{U}$ to accomplish a specific task. Driven by u_k , the system transitions from state x_k to a subsequent state x_{k+1} according to discrete dynamics D , thereby closing the loop. The system's evolution over a single step depends on both the current state x_k and the environment status e_k , and is expressed as:

$$x_{k+1} = D(x_k, C(P(x_k, e_k))). \quad (1)$$

Definition 2 (System Evolution) The system starts from an initial state x_0 within a set $\mathcal{I} \subseteq \mathcal{X}$. At each time step i between 0 and $k - 1$, the corresponding environment is represented as $e_i \subseteq \mathcal{E}$. Consequently, the system state x_k evolves from x_0 by unrolling the dynamics defined in Eq. (1). This system evolution depends on the initial state x_0 and the series of the environment parameters e_0, \dots, e_{k-1} over time:

$$x_k = \text{System}(x_0, \{e_0, \dots, e_i, \dots, e_{k-1}\}). \quad (2)$$

Definition 3 (Reachable Set) The reachable set at step k is

$$R_k = \{\text{System}(x_0, \{e_0, \dots, e_{k-1}\}) \\ | \forall x_0 \in \mathcal{I}, \forall_{i \in [0, k-1]} e_i \in \mathcal{E}\}.$$

Here, R_0 denotes the reachable set at time step 0, which is equal to the initial set \mathcal{I} . Furthermore, the reachable set over the time interval $[0, k]$ is defined as $R_{[0, k]} = \bigcup_{i \in [0, k]} R_i$.

Problem 1 (Safety Verification for Image-Based NNCS) The objective of the safety verification for image-based NNCS is to check whether the computed reachable set adheres to the system's safety property. Formally, given an unsafe region $\mathcal{U} \subseteq \mathcal{X}$, the safety of the NNCS within a bounded time step k_{\max} can be verified if and only if the following condition is satisfied, that is $R_{[0, k_{\max}]} \cap \mathcal{U} = \emptyset$.

Perception System Approximation

One of the most significant challenges in verifying image-based NNCS is formalization of the perception system. Consider a camera used for autonomous driving. Formalizing a specification like “all images with a car that is 5 meters ahead are predicted as such” is difficult due to the interdependence of the captured images on multiple factors, including both the system state x (e.g., distance between the ego car and the leading car) and environment e (e.g., appearance of the car, weather condition, and other objects on the road). The challenge lies in providing a clear mathematical and comprehensive description for such a specification.

An alternative approach (Katz et al. 2022) involves using images generated by a cGAN to replace those rendered by the actual perception system for verifying the image-based NNCS. In this context, the specification can be formalized as “all images that can be produced by a specific cGAN are predicted to be a car 5 meters ahead.” Such a specification is practical to work with, and in this paper, we embrace this methodology to establish the verification problem for the closed-loop system. Of course, safety of the cGAN does not guarantee safety in the real world, but this approach does offer an analysis method for this class of systems beyond just simulations and tests. The cGAN learns the distribution of images conditioning on auxiliary information and

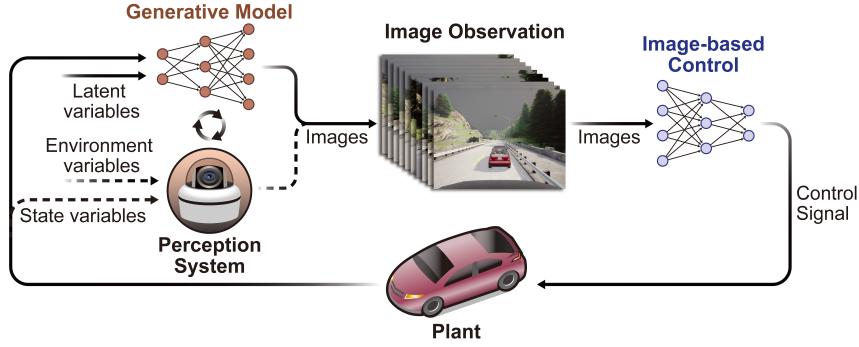


Figure 1: Simplified architecture of an image-based neural network control system and its surrogate system.

can be used to approximate the original perception system P into a *surrogate perception system* \hat{P} . The generator of the cGAN maps the conditional information c and a set of latent variables z into an image observation \hat{o} . When the cGAN serves as a perception surrogate, the conditional information c corresponds to the system states x , guiding the generator to create images relevant to the desired information. The environmental variables e are left uncontrolled and represented by latent variables z within the cGAN.

Definition 4 (Surrogate System Model) *The one-step system evolution with the surrogate perception system is $x_{k+1} = D(x_k, C(\hat{P}(x_k, e_k)))$.*

Problem 2 (Verification for Surrogate System) *Instead of verifying the actual system as defined in Problem 1, this paper focuses on the verification of the surrogate system.*

Verification Challenges

The concatenation of the cGAN and the image-based controller into a unified neural network transforms the entire system into a state-based NNCS. In this way, the image-based NNCS can be analyzed using existing methodologies designed for verifying state-based NNCS (Xiang and Johnson 2018). However, computing reachable sets for an NNCS can still be challenging, especially given the complexity of the network performing both image generation and processing tasks. In contrast, in the most recent NNCS verification competition, ARCH-COMP 2023 (Lopez et al. 2023), the analyzed neural networks generally had only a few dozen neurons and up to 6 layers. Most conventional NNCS verification tools (Bogomolov et al. 2019), including those participating the competition, lack support for convolutional networks. Even those that do offer support (Tran et al. 2020), encountering large-scale networks leads to significant overapproximations at each step. These overapproximations accumulate and expand, causing the tool to fail.

In earlier work, to practically address the verification of the surrogate image-based NNCS, input images are down-sized to small grayscale images, and the cGAN and controller are implemented using feed-forward neural networks. Still, this method exhibits large overapproximation of the reachable sets. Such substantial overapproximation could result in instances where the overapproximated reachable sets

intersect with unsafe regions, causing safety verification to fail. As contemporary image-based control systems grow in complexity, with high-resolution images and advanced architectures like convolutional and transformer layers, verification becomes even more challenging. **The main contribution of this paper is to address these challenges and reduce the overapproximation error.**

Methodology

This section analyzes the causes of overapproximation in the prior method and proposes a method to mitigate this issue.

Overapproximation Analysis

Consider Problem 2, where a surrogate image-based NNCS starts from an initial set R_0 . The verification task is to determine whether the reachable set up to time k_{\max} satisfies the safety property, i.e., avoiding any intersection with the unsafe set \mathcal{U} . Neural network verification tools are commonly employed to prove properties over the network's input and output. Therefore, if the system's evolution function, as defined by Eq. (2), is combined into a unified neural network NN, the closed-loop system property can be verified using network verification tools. Formally, the set of possible outputs of network (system state at step k_{\max}) is:

$$\text{Range}(\text{NN}, R_0, \mathcal{E}^{k_{\max}}) = \{\text{NN}(x_0, \{e_0, \dots, e_{k_{\max}-1}\}), \\ | \forall x_0 \in R_0, \forall_{i \in [0, k_{\max}-1]} e_i \in \mathcal{E}\}.$$

The neural network verification problem is to check if $\text{Range}(\text{NN}, R_0, \mathcal{E}^{k_{\max}}) \cap \mathcal{U} = \emptyset$. However, this task presents challenges. First, system dynamics often involve nonlinear operations that may not be supported by verification tools. Second, these tools may have scalability limitations in that large input sets and complex neural networks may present intractable verification problems. Unrolling the system evolution function to span from time step 0 to k_{\max} replicates a sequence of cGANs, image-based controllers, and dynamics layers. This composed network can be too complex even for state-of-the-art neural network verification tools.

To address these challenges, a practical approach integrating neural network verification tools with reachability methods was proposed in previous works (Xiang and Johnson 2018), later extended to verify the surrogate image-based

NNCS (Katz et al. 2022). This method serves as a baseline for comparison in this paper. The prior approach constructs and analyzes a discrete existential abstraction of the system, with transitions defined using one-step reachability. The process starts by dividing the state space into a finite number of rectangular cells along a grid \mathcal{H} . The initial states in R_0 are then abstracted using a set of rectangular cells referred to as \mathcal{C}_0 , where $R_0 \subseteq \mathcal{C}_0 \subseteq \mathcal{H}$. For each cell c within \mathcal{C}_0 , the interval bounds on the control commands are determined using a neural network verification tool. These bounds are combined with a monotonic analysis of the system dynamics to result in an interval of possible one-step successors from state cell c , called \mathcal{R}_1^c . For example, in the autonomous aircraft taxiing system we will analyze later, the dynamics updating function for the heading angle error θ is: $\theta_{k+1} = \theta_k + \frac{v}{L} \Delta t \tan \phi_k$, where v , L and Δt are constants, ϕ_k is the control signal. As the tan function is monotonically increasing in the specified operating region, monotonic analysis computes the upper (lower) bound of θ_{k+1} by simultaneously substituting the upper (lower) bounds for both θ_k and ϕ_k into the dynamics function (Katz et al. 2022).

The set of possible successor states from c is then the set of cells \mathcal{C}_1^c that overlap with \mathcal{R}_1^c . An overapproximation of the reachable set for the entire initial state is then the union of all successors from individual cells, $\mathcal{C}_1 = \bigcup_{c \in \mathcal{C}_0} \mathcal{C}_1^c$, which overapproximates $\mathcal{R}_1 = \bigcup_{c \in \mathcal{C}_0} \mathcal{R}_1^c$, and the exact reachable set R_1 , denoted as $R_1 \subseteq \mathcal{R}_1 \subseteq \mathcal{C}_1$. This process is repeated iteratively to obtain reachable set \mathcal{C}_k for arbitrary time step k . The first row of Fig. 2 illustrates this computation flow from R_0 to \mathcal{C}_1 , and the first row of Fig. 3 displays the resulting reachable sets from R_0 to \mathcal{C}_2 . Note that the latent space representing the environment is not divided, and thus, the entire set \mathcal{E} is considered when analyzing each cell.

However, this baseline method introduces considerable overapproximation error, which we call *one-step error* and *multi-step error*. The one-step error reflects the discrepancy between the overapproximated reachable set \mathcal{R} and the exact reachable set R within a single step. This error arises from two factors within the baseline method. First, the method computes the interval of the control outputs without accounting for the input-output (state-control) dependencies; second, the method uses monotonic analysis for system dynamics, resulting in an interval enclosure of the reachable set for the next state, which is a coarse overapproximation of the exact reachable set. The one-step error is illustrated in the middle of the first row in Fig. 3, where, at step 1, the exact reachable set is a gray curvilinear triangle, while the baseline algorithm overapproximates it as a pink rectangle. The multi-step error, on the other hand, arises from the abstraction error when transitioning from \mathcal{R} to \mathcal{C} . As shown in the top middle in Fig. 3, after overapproximating the reachable set with the pink rectangle, the baseline method further abstracts it with 4 blue boxes, introducing the multi-step error. As steps increase, both of errors interact and accumulate, compromising the verification accuracy. At time step 2 (top right, Fig. 3), with the baseline method, all 9 cells in the graph are considered reachable, while the exact reachable set occupies a curvilinear triangle smaller than 3 cells.

Proposed Improvements

Composition Composing the system dynamics with the cGAN and controller within a single control period can preserve the dependencies between input states and control outputs, thereby reducing the one-step error. This computation flow is depicted in the second row of Fig. 2.

The first proposed method for composition is to directly append the system dynamics to the neural networks as an additional layer. Verification tools can then assess the output specifications of a complete step given the input sets, internally preserving the state-control dependencies. Rather than computing interval bounds on the control outputs, the tools would be used to compute interval bounds on the successor states. However, there are drawbacks to this composition method. First, it assumes that the system dynamics are either linear or consist only of nonlinear operations supported by a neural network verification tool. Even if supported, verification tools may still fail to produce bounds. For instance, many state-of-the-art neural network verification tools, like α, β -CROWN (Zhang et al. 2018), are based on bounds propagation and refinement. Nonlinear functions like tan, while supported, might have infinite outputs when their ranges are overapproximated—if the input to tan is initially overapproximated beyond $(-\frac{\pi}{2}, \frac{\pi}{2})$. In addition, although this composition can result in a tighter interval enclosure of the reachable set compared to the baseline method, it is still not exact and can cause some one-step error.

A second composition approach involves combining dependency-preserving exact neural network analysis methods with reachability analysis algorithms from hybrid systems. Unlike bounds propagation methods, dependency-preserving methods such as those used in nnenum (Bak 2021) or NNV (Tran et al. 2020) provide the set of possible outputs along with their relationship to inputs, represented as a union of star sets (Duggirala and Viswanathan 2016). (sometimes called constrained zonotopes (Scott et al. 2016)). When dynamics are nonlinear, reachability analysis algorithm, specifically the conservative linearization approach (Althoff, Stursberg, and Buss 2008), is used. This method soundly overapproximates nonlinear systems as locally linear with added noise to account for the maximum linearization error. While this approach introduces some overapproximation from the linearization error, the reachable set is significantly more accurate than the monotonic analysis approach in the prior work, reducing one-step error. Additionally, this composition method does not have the infinite output problem when the dynamics involve tan or other reciprocal functions. This composition method does have scalability limitations related to the size and type of the neural network, as exact analysis of the network is required.

Unrolling Multi-step error is attributed to the abstraction operation from reachable set \mathcal{R} to a set of cells \mathcal{C} at each step. To reduce the frequency of abstractions and, consequently, the multi-step error, we propose to unroll multiple control periods and combine them into a single large neural network. This strategy composes multiple steps of analysis into a single larger operation, to avoid error caused by repeated abstraction step. The third row of Fig. 2 illustrates

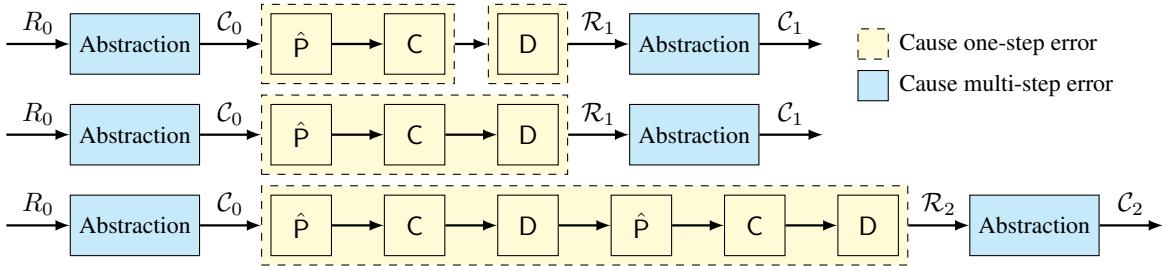


Figure 2: Computation flowcharts for the baseline (first row), 1-step (second row), and 2-step (third row) methods.

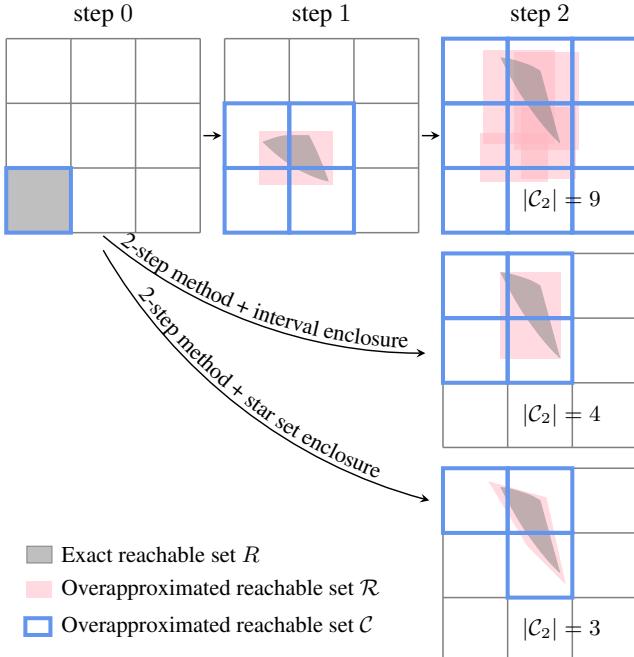


Figure 3: Illustrations of reachable sets using three different methods. In this illustrative example, assume the initial exact reachable set R_0 only contains one cell c , $R_0 = \mathcal{C}_0 = \{c\}$.

this strategy with a 2-step unrolling. Increasing the number of unrolling steps m generally leads to a smaller overapproximation error, but a more complex verification problem at each step. Thus, the selection of m depends on both the system complexity and the verification tool used.

Improvement Illustration The composition and unrolling strategies collectively contribute to reducing overapproximation, as illustrated in the second and third rows of Fig. 3. The second row shows the method employing a 2-step unrolling strategy, coupled with the first composition approach wherein the dynamics are integrated as a network layer. This approach enables direct computation of the interval enclosure for reachable set at step 2, which is then abstracted into 4 cells, as opposed to the 9 cells in baseline method shown in the first row. The third row illustrates the method with a 2-step unrolling strategy, while using the composition based on exact neural network analysis and reachability algorithm for the dynamics. The reachable set at

step 2 is represented by star sets, which aligns more closely with the exact reachable set. By checking the intersection between the star sets and rectangular cells (Wetzlinger et al. 2023), the reachable set at time step 2 can be reduced to 3 cells compared to 4 cells using the interval enclosure.

Overall Algorithm Here we introduce the overall algorithm for the reachability analysis, with the detailed pseudocode presented in Algorithm 1, and a formal justification are provided in Appendix A. We start by initializing the global reachable set \mathcal{C} with a set of cells \mathcal{C}_0 abstracted from the initial set R_0 . The abstraction process is formally expressed as $\mathcal{C} = \alpha(\mathcal{R}, \mathcal{H})$, where \mathcal{H} is the set of all rectangular cells in the state space. Assuming the number of unrolling steps as m , the reachable set after m steps \mathcal{R}_{k+m} from time step k is denoted as $\mathcal{R}_{k+m} = \text{ComputeReach}_m(\mathcal{C}_k)$. It should note that the computation process must be applied for each cell within \mathcal{C}_k , with results then being combined to obtain the reachable set. However, using ComputeReach_m alone allows us to compute reachable sets only when the time steps are multiples of m . To ensure soundness we must span the entire time domain, requiring ComputeReach_1 through $\text{ComputeReach}_{m-1}$. Starting from \mathcal{C}_0 , we compute reachable sets \mathcal{R}_1 through \mathcal{R}_m , which are then abstracted into \mathcal{C}_1 to \mathcal{C}_m . These newly computed sets are accumulated into the global reachable set \mathcal{C} . The reachable set \mathcal{C}_m is used as the starting set for the next iteration. As \mathcal{C}_m serves an overapproximation of the exact reachable set R_m , computing the successor reachable sets from \mathcal{C}_m ensures the soundness of the reachability analysis. The iterative process continues until either the reachable set has converged to an invariant set, or the time step reaches the bounded step k_{\max} . Finally, the algorithm returns the global reachable set \mathcal{C} and a safety flag indicating if \mathcal{U} was reached. We additionally introduce a backward reachability algorithm designed to identify all cells that can be guaranteed to be safe. This algorithm is comprehensively outlined in Appendix A.

Case Studies

Autonomous Aircraft Taxiing System

We use the autonomous aircraft taxiing system, evaluated by the baseline method, to demonstrate our improvements.

System Details The system’s task is to control the steering of an aircraft moving at a constant speed on a taxiway according to nonlinear discrete dynamics: $p_{k+1} = p_k +$

Algorithm 1: Proposed reachability algorithm.

```

1 Function ReachAnalysis:
2   Input:  $R_0$ , initial set
3   Input:  $\mathcal{U}$ , unsafe region
4   Input:  $m$ , unrolling steps
5   Input:  $k_{\max}$ , termination time step
6   Input:  $\alpha$ , abstraction function
7   Input:  $\mathcal{H}$ , rectangular cells defined within the
8     state space
9
10   $\mathcal{C} := \mathcal{C}_0 := \alpha(R_0, \mathcal{H})$ 
11   $k := 0$ 
12  isConverged := false
13  while  $\neg$  isConverged and  $k < k_{\max}$  do
14    /* compute reach set for  $k + 1$ 
15      to  $k + m$  */  

16    for  $i = 1$  to  $m$  do
17       $\mathcal{R}_{k+i} := \text{ComputeReach}_i(\mathcal{C}_k)$ 
18       $\mathcal{C}_{k+i} := \alpha(\mathcal{R}_{k+i}, \mathcal{H})$ 
19       $\mathcal{C} := \mathcal{C} \cup \mathcal{C}_{k+i}$ 
20    end
21    isConverged := ( $\mathcal{C}_k = \mathcal{C}_{k+m}$ )? true : false
22     $k := k + m$ 
23  end
24  isSafe := ( $\mathcal{C} \cap \mathcal{U} = \emptyset$ )? true : false
25  return  $\mathcal{C}$ , isSafe

```

$v\Delta t \sin \theta_k$ and $\theta_{k+1} = \theta_k + \frac{v}{L}\Delta t \tan \phi_k$. The aircraft's state is defined by its crosstrack position p and heading angle error θ . Here, v , Δt , and L represent the aircraft's taxi speed (5 m/s), dynamics updating period (0.05 s), and the distance between front and back wheels (5 m), respectively. The control signal ϕ_k is generated by an image-based controller running at 1 Hz, which first uses a neural network to predict the state variables \hat{p}_k and $\hat{\theta}_k$, followed by a proportional control strategy: $\phi_k = -0.74\hat{p}_k - 0.44\hat{\theta}_k$. The perception is approximated using a cGAN conditioned on the aircraft's states p and θ . Two latent variables from -0.8 to 0.8 are introduced to capture environment variations. The unified network, which includes both the cGAN and the image-based controller, consists of 8 fully-connected layers with ReLU activations. Further details on the network architecture and the generated images are available in the Appendix B.

Verification Results The state space is defined with $p \in [-11 \text{ m}, 11 \text{ m}]$ and $\theta \in [-30^\circ, 30^\circ]$. Consistent with prior work, we partition the space into a grid of 128×128 cells with uniform cell width in each dimension.

We initiate our comparison by contrasting the proposed method with the baseline through a single-cell reachability analysis. The proposed approach begins by composing the network and the dynamics to preserve the dependencies between input states and control outputs. We first try to integrate the system dynamics as an additional layer within the neural networks. Despite support for \sin and \tan within α, β -CROWN, the verification process *fails* in our experiment due to the infinite output of the \tan function during the initial refinement round, as discussed in Methodology section. We next try the proposed alternative composition

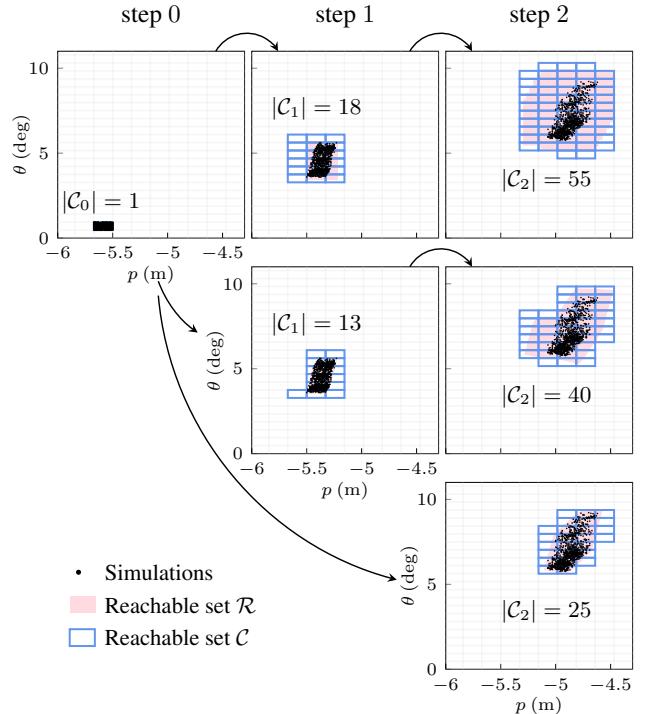


Figure 4: Reachable sets computed using baseline (first row), 1-step (second row), and 2-step (third row) methods in the taxiing system. The initial set only contains one cell.

approach, using exact analysis with nnenum and reachability analysis. To account for the nonlinear dynamics, we employ a conservative linearization technique based on Taylor expansion (Althoff, Stursberg, and Buss 2008). Detailed explanations of this technique are provided in Appendix B.

This reachability method tracks dependencies between the initial state and the next state under a control command, and therefore has less overapproximation compared to monotonic analysis, as illustrated in Fig. 4. Starting from the same single cell (row 1, column 1), the pink reachable set in the proposed method (row 2, column 2), exhibits a smaller area compared to the pink box interval estimated by the baseline (row 1, column 2). Furthermore, the number of abstract successor states is reduced from 18 in the baseline to 13 when dependency-preserving composition is used.

The multi-step unrolling strategy repeats the composition and propagates the resulting star sets across multiple steps. As shown in Fig. 4, the 2-step method (row 3, column 3) at step 2 significantly reduces overapproximation compared to the 1-step method (row 2, column 3), primarily by mitigating multi-step error. Consequently, comparing the baseline, 1-step, and 2-step methods at step 2, the number of reachable cells decreases from 55 to 40 to 25, with reachable sets becoming closer to simulation results. As the number of steps m increases, the verification task become more complex to the growing scale of network architectures and expanding input dimensions with two additional latent variables at each step. In this experiment, we consider $m = 1, 2$.

Two properties are evaluated for the closed-loop system:

- P1: The aircraft remains on the runway, ensuring that the

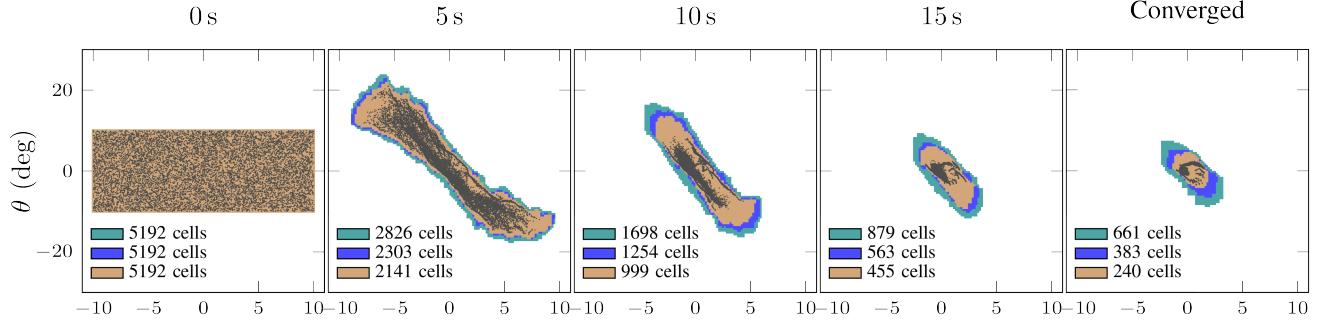


Figure 5: Reachable sets over time using three different methods when starting from $p \in [-10 \text{ m}, 10 \text{ m}]$ and $\theta \in [-10^\circ, 10^\circ]$ in taxiing system. The colors black, teal, blue, and brown represent simulations, baseline, 1-step, and 2-step methods, respectively.

- magnitude of the crosstrack error does not exceed 10 m.
- P2: The aircraft is steered toward the runway center, leading to convergence of the reachable set to an invariant set.

We begin by evaluating P2 using the same initial set as prior work, where $p \in [-10 \text{ m}, 10 \text{ m}]$ and $\theta \in [-10^\circ, 10^\circ]$. The simulations and reachable sets computed using the baseline, 1-step, and 2-step methods are depicted in Fig. 5. Over time, the reachable sets of all three methods gradually contract and eventually converge to invariant sets at 22 s, 22 s, and 26 s, respectively. This demonstrates P2, showing that the aircraft, guided by the image-based controller, converges near the runway center. Further, the reachable sets computed using the baseline, 1-step, and 2-step methods progressively decrease and align more closely with the simulations. Upon convergence, the number of reachable cells for three methods are 661, 383, and 240, respectively. The steady state area with the baseline method is 175% larger than our two-step approach. These numerical results show that the proposed methods significantly reduce the overapproximation compared to the baseline. The importance of our accuracy improvements becomes more apparent when parameters of the case study are altered, detailed in the Appendix B.

Advanced Emergency Braking System

We next evaluate the proposed approach using an advanced emergency braking system (Cai and Koutsoukos 2020) with an image-based controller, demonstrating that our improvements permit the verification of image-based NNCS that are significantly more complex than the prior work.

System Details The system is to apply braking force to safely stop the host vehicle when approaching a stopped vehicle ahead. The state is defined by the distance to the obstacle d and the host vehicle's velocity v , evolving according to linear dynamics: $d_{k+1} = d_k - v_k \Delta t$ and $v_{k+1} = v_k - a_k \Delta t$, where time step $\Delta t = 0.05 \text{ s}$. The deceleration a_k is calculated as $a_k = 0.009u_k + 0.0042$, with u_k representing the braking force predicted by an image-based controller.

We consider two versions of an image-based controller: one using a convolutional network and the other a transformer architecture. The perception system is abstracted using a cGAN, which incorporates the distance d as a conditional variable, along with four latent variables, each

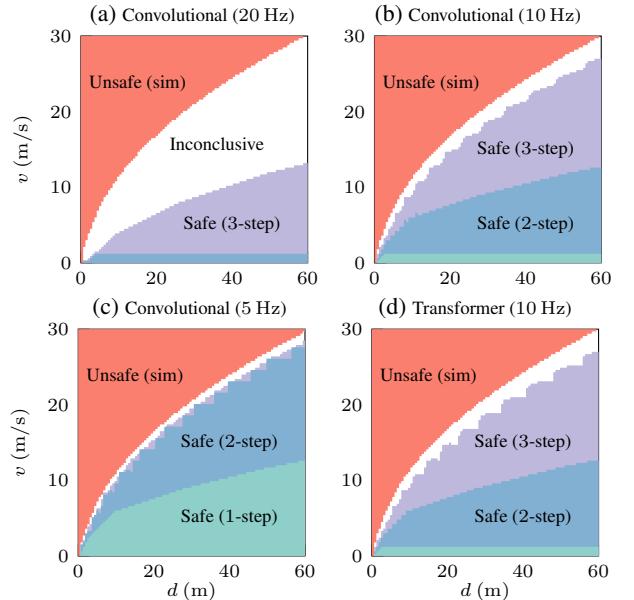


Figure 6: The state sets guaranteed to be safe for different controllers in braking system. Cells are identified as unsafe by simulations are colored in red, and cells are verified as safe using 1-, 2-, and 3-step methods are in green, blue, and purple. Quantitative results are presented in Appendix C.

bounded by $\pm 10^{-2}$. The cGAN also comes in two network variants, aligning with the controller versions. The convolutional variant comprises a total of 8 convolutional layers, including both cGAN and controller, while the transformer variant includes a total of 22 convolutional layers and 2 self-attention layers. Detailed network architectures and the generated images are available in Appendix C.

Verification Results The state space with $d \in [0 \text{ m}, 60 \text{ m}]$ and $v \in [0 \text{ m/s}, 30 \text{ m/s}]$ is divided into a grid of 100×100 equal-size cells, while the latent space is not divided. Given the extensive scale and the non-ReLU activation functions in the network, performing composition with the exact analysis approach is infeasible. Instead, we perform composition the other way, by appending the linear dynamics to the neural networks as an additional layer and then running range anal-

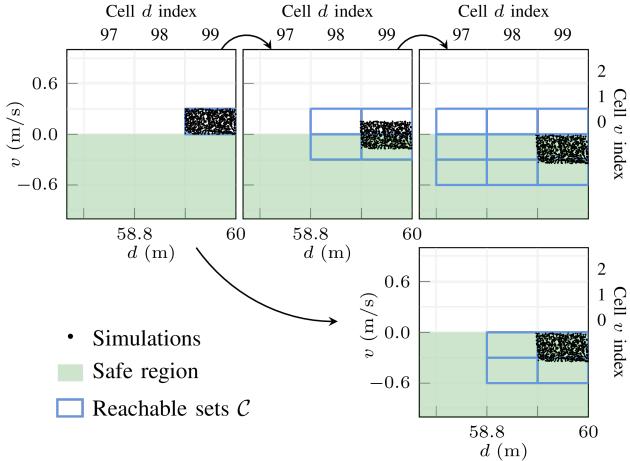


Figure 7: Comparison of the reachable sets from a single cell computed by 1-step (first row) and 2-step methods (second row) in the braking system.

ysis with $\alpha\text{-}\beta$ -CROWN. Since the state space is pre-divided, the process of abstracting reachable sets into cells can be integrated into the verification process. For each output, we begin with simulations to determine the minimized range of the output, then identify the lower and upper indices of cells where all simulations are located. We incrementally expand the indices until the output is proved to be safe within the range. This approach allows us to effectively compute the reachable set represented by interval bounds.

For this system, we consider a single specification:

- P1: The host vehicle comes to a stop before colliding with the lead, meaning that the distance to the leading vehicle should never reach 0 m before the velocity reaches 0 m/s.

First, we analyze the image-based controller using the convolutional network variant. We run 5000 simulations starting from random points in each cell, in order to estimate the number of possible safe cells. In Fig. 6a, 3463 cells within the state space are detected as unsafe using simulations, while the remaining cells are candidate safe cells. We unroll the system with $m = 1, 2$, and 3 steps, and run backward reachability analysis to show which states can be proven safe in Fig. 6a. Surprisingly, no states can be proven safe using the 1-step method, even in situations with large distances and nearly-zero velocities. Compared with the aircraft taxiing system, the dynamics in this system moves the states significantly less at each control cycle, leading to excessive multi-step error from the abstraction process. This is illustrated and detailed discussed in the first row of Fig. 7. Starting from cell $(99, 0)$, the 1-step method extends the reachable set a cell located on the left, $(98, 0)$ after one step. This trend continues, eventually extending reachable set to the leftmost cell where an unsafe state is reached. This experiment also demonstrates that the baseline method is ineffective in verifying such a system, given the 1-step method has lower overapproximation compared to the baseline approach. However, as shown in the second row, the 2-step method can prove the safety of cell $(99, 0)$.

As the unrolling steps m increases, the number of prov-

ably safe cells also increases, with 2-step and 3-step methods verifying 384 and 2669 cells, respectively. However, a substantial number of 3868 cells remain inconclusive. This occurs due to the control cycle’s period of 0.05 s, where even 3-step analysis has substantial multi-step error. Unrolling beyond 3 steps becomes practically difficult as the composed neural network increases in size, leading to an increase analysis time with the $\alpha\text{-}\beta$ -CROWN verification tool. Detailed results of runtime is provided in the Appendix D.

As the primary cause of inconclusive cells is multi-step error from the abstraction process, we can consider decreasing the control frequency to reduce this effect. This is counterintuitive—a lower frequency should lead to *worse* control performance and therefore more unsafe cells. However, due to reduced multi-step error this could actually allow the method to verify more cells as provably safe. We analyze the system by reducing the control frequency from 20 Hz to 10 Hz (Fig. 6b) and 5 Hz (Fig. 6c). As expected, the control performance becomes slightly worse as the control frequency decreases, resulting in a minor increase in the number of unsafe cells identified through simulation, by 1.7% for 10 Hz and 4.4% for 5 Hz. However, the decrease in frequency improves the ability to establish provably safe cells for all methods compared to their original results—the white gap in the figure is reduced from 38.7% of the cells with the original 20 Hz frequency. For the 10 Hz, with 3-step analysis only 7.4% of the cells remain inconclusive. For the 5 Hz, this is further reduced to 3.5% of the cells.

We also evaluate the controller with transformer layers (Fig. 6d), where simulations identify 3517 unsafe cells. As m increases, the numbers of verified safe cells increase, reaching 390, 2890, and 5707 using 1-, 2-, and 3-step methods, respectively, while 7.8% of cells remain inconclusive.

Conclusions

Analysis of image-based NNCS typically involves running lots of tests. By leveraging generative models to approximate the perception system, the surrogate verification approach offers an alternative that can utilize computational set-based analysis methods. Scalability is the primary bottleneck in this line of work and the main problem addressed in this paper. We identified underlying causes of overapproximation error with the existing baseline method and proposed two strategies—composition and unrolling—to overcome these problems. Our evaluation demonstrates the advantages of our approach over the baseline. In terms of accuracy, our method reduces the converged reachable set by 175% on the aircraft taxiing system compared to the baseline. In terms of scalability, the transformer variant of the cGAN in emergency braking case study has $24\times$ more output pixels than the prior work’s case study.

Despite improvements, scalability is not yet solved, as state-of-the-art generative models complex and continue to grow in size. Increasing the ranges and number of latent variables, generating higher-resolution images, considering systems with more state variables and incorporating recent video GAN techniques (Tulyakov et al. 2018) will all require further improvements to scalability.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research and the Office of Naval Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, FA9550-23-1-0066 and N00014-22-1-2156, and the National Science Foundation under Award No. 2237229. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force or the United States Navy.

References

- Althoff, M.; Stursberg, O.; and Buss, M. 2008. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *IEEE Conference on Decision and Control*.
- ArjomandBigdeli, A.; Mata, A.; and Bak, S. 2024. *Verification of Neural Network Control Systems in Continuous Time*, 100–115. Springer Nature Switzerland. ISBN 9783031651120.
- Astorga, A.; Hsieh, C.; Madhusudan, P.; and Mitra, S. 2023. Perception contracts for safety of ml-enabled systems. *Proceedings of the ACM on Programming Languages*, 7(OOPSLA2): 2196–2223.
- Bak, S. 2021. nnenum: Verification of ReLU Neural Networks with Optimized Abstraction Refinement. In *NASA Formal Methods Symposium*.
- Bogomolov, S.; Forets, M.; Frehse, G.; Potomkin, K.; and Schilling, C. 2019. JuliaReach: a toolbox for set-based reachability. In *ACM International Conference on Hybrid Systems: Computation and Control*.
- Boloor, A.; Garimella, K.; He, X.; Gill, C.; Vorobeychik, Y.; and Zhang, X. 2020. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture*.
- Cai, F.; and Koutsoukos, X. 2020. Real-time Out-of-distribution Detection in Learning-Enabled Cyber-Physical Systems. In *ACM/IEEE International Conference on Cyber-Physical Systems*.
- Cai, F.; Li, J.; and Koutsoukos, X. 2020. Detecting adversarial examples in learning-enabled cyber-physical systems using variational autoencoder for regression. In *IEEE Security and Privacy Workshops*.
- Chen, C.; Seff, A.; Kornhauser, A.; and Xiao, J. 2015. Deep-driving: Learning affordance for direct perception in autonomous driving. In *IEEE international conference on computer vision*.
- Duggirala, P. S.; and Viswanathan, M. 2016. Parsimonious, simulation based verification of linear systems. In *International Conference on Computer Aided Verification*.
- Habeeb, P.; Deka, N.; D’Souza, D.; Lodaya, K.; and Prabhakar, P. 2023. Verification of camera-based autonomous systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(10): 3450–3463.
- Hsieh, C.; Li, Y.; Sun, D.; Joshi, K.; Misailovic, S.; and Mitra, S. 2022. Verifying controllers with vision-based perception using safe approximate abstractions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11): 4205–4216.
- Jaulin, L.; Kieffer, M.; Didrit, O.; Walter, E.; Jaulin, L.; Kieffer, M.; Didrit, O.; and Walter, É. 2001. *Interval analysis*. Springer.
- Katz, S. M.; Corso, A. L.; Strong, C. A.; and Kochenderfer, M. J. 2022. Verification of image-based neural network controllers using generative models. *Journal of Aerospace Information Systems*.
- Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; and Quillen, D. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International journal of robotics research*.
- Lopez, D. M.; Althoff, M.; Forets, M.; Johnson, T. T.; Ladner, T.; and Schilling, C. 2023. ARCH-COMP23 Category Report: Artificial Intelligence and Neural Network Control Systems (AINNCS) for Continuous and Hybrid Systems Plants. *EPiC Series in Computing*.
- Mirza, M.; and Osindero, S. 2014. Conditional Generative Adversarial Nets. *arXiv preprint*.
- Păsăreanu, C. S.; Mangal, R.; Gopinath, D.; Getir Yaman, S.; Imrie, C.; Calinescu, R.; and Yu, H. 2023. Closed-loop analysis of vision-based autonomous systems: A case study. In *International conference on computer aided verification*.
- Schilling, C.; Forets, M.; and Guadalupe, S. 2022. Verification of neural-network control systems by integrating Taylor models and zonotopes. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Scott, J. K.; Raimondo, D. M.; Marseglia, G. R.; and Braatz, R. D. 2016. Constrained zonotopes: A new tool for set-based estimation and fault detection. *Automatica*, 69: 126–136.
- Sun, X.; Khedr, H.; and Shoukry, Y. 2019. Formal verification of neural network controlled autonomous systems. In *ACM International Conference on Hybrid Systems: Computation and Control*.
- Tran, H.-D.; Yang, X.; Manzanas Lopez, D.; Musau, P.; Nguyen, L. V.; Xiang, W.; Bak, S.; and Johnson, T. T. 2020. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*.
- Tulyakov, S.; Liu, M.-Y.; Yang, X.; and Kautz, J. 2018. Mocogan: Decomposing motion and content for video generation. In *IEEE conference on computer vision and pattern recognition*, 1526–1535.
- Wetzlinger, M.; Kochdumper, N.; Bak, S.; and Althoff, M. 2023. Fully Automated Verification of Linear Systems Using Inner-and Outer-Approximations of Reachable Sets. *IEEE Transactions on Automatic Control*.
- Xiang, W.; and Johnson, T. T. 2018. Reachability analysis and safety verification for neural network control systems. *arXiv preprint*.
- Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2019. Self-attention generative adversarial networks. In *International conference on machine learning*.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*.

A Appendix for Methodology

A.1 Forward Reachability Algorithm

Completeness Justification: The forward reachability algorithm iteratively computes the reachable set using the functions `ComputeReach` and the abstraction process α . The completeness of the algorithm is ensured by the properties of these components. The abstraction process α identifies the set of cells \mathcal{C} that overlap with the reachable region \mathcal{R} , providing an overapproximation of the exact reachable set R . This guarantees that all reachable states are included, ensuring the abstraction process captures the entire reachable region. The computation of reachable sets through `ComputeReach` is complete, as it relies on two composition options whose completeness is guaranteed by their underlying algorithms. The first option uses neural network verification tools with proven completeness guarantees, while the second leverages reachability algorithms from hybrid systems, which are also formally complete.

Convergent Justification: The forward reachability algorithm is guaranteed to converge, as it terminates either when the reachable set stabilizes or when the predefined maximum time step k_{\max} is reached.

A.2 Backward Reachability Algorithm

The algorithm utilizes the backward reachability analysis to determine a set of cells \mathcal{A} within the state space that may potentially lead to unsafe system states, and the set of cells that can be guaranteed to be safe is essentially the complement of set \mathcal{A} . To streamline the process, the function of computing the reachable set (`ComputeReach`) and the abstraction process (α) can be combined into a single function `ComputeReachCells`. This function calculates the forward reachable cells starting from a single cell or a set of cells. Given that the input space is discretized into a finite number of cells, evaluating the forward reachable cells for each individual cell enables the construction of a reverse mapping `ComputeBackReachCells`, which can identify all cells that can reach a target cell or a set of target cells.

The algorithm for backward reachability analysis begins by initializing the set of potentially unsafe cells \mathcal{A} . This is done by considering the cells themselves as unsafe (line 2) or identifying cells that can reach an unsafe state through propagation using any one of the methods ranging from 1-step to $(m - 1)$ -step approaches (lines 3-10). For each cell c that is newly added to \mathcal{A} , the algorithm computes all cells that could reach c using an m -step backward reachability analysis. These newly identified cells are considered potentially unsafe and are added to the set \mathcal{A} . This iteration continues until no new cells are added to \mathcal{A} , and the algorithm returns the set of possible unsafe cells \mathcal{A} within the state space.

Completeness Justification: If a cell is unsafe, it must reach an unsafe region at some step k in the future. Consider $k // m = a$ and $t \% m = b$. This cell must be included in the set computed using the following procedure defined within the algorithm. This set is obtained by first finding the set of cells that can reach an unsafe state using a b -step method (initialization procedure in the algorithm) and then back-

Algorithm 2: Proposed backward reachability algorithm.

```

1 Function BackReachAnalysis:
    Input:  $\mathcal{U}$ , unsafe region
    Input:  $m$ , unrolling steps
    Input:  $\mathcal{H}$ , rectangular cells defined within the
          state space
    /* initialize the possible unsafe
       set */
     $\mathcal{A} := \mathcal{U} \cap \mathcal{H}$ 
    foreach  $c \in \mathcal{H}$  do
        for  $i = 1$  to  $m - 1$  do
             $\mathcal{C} := \text{ComputeReachCells}_i(c)$ 
            if  $\mathcal{C} \cap \mathcal{U} \neq \emptyset$  then
                |  $\mathcal{A} := \mathcal{A} \cup \{c\}$ 
            end
        end
    end
    /* Compute all possible unsafe
       cells using  $m$ -step backward
       reachability */
     $\mathcal{N} := \mathcal{A}$ 
    while  $\mathcal{N} \neq \emptyset$  do
         $\mathcal{N}' := \emptyset$ 
        foreach  $c \in \mathcal{N}$  do
            |  $\mathcal{C} := \text{ComputeBackReachCells}_m(c)$ 
            |  $\mathcal{N}' := \mathcal{N}' \cup \mathcal{C}$ 
        end
         $\mathcal{N} := \mathcal{A} \cap \mathcal{N}'$ 
         $\mathcal{A} := \mathcal{A} \cup \mathcal{N}'$ 
    end
return  $\mathcal{A}$ 

```

propagating this set a times using an m -step method (iteration procedure in the algorithm).

Convergent Justification: Since the state space is quantized into a finite number of cells, it is impossible to add new cells to the set \mathcal{A} infinitely.

B Appendix for Autonomous Aircraft Taxiing System

B.1 Network Architecture and Generated Images

In the prior work with the baseline method (Katz et al. 2022), several simplifications were made to make the verification of the surrogate system tractable. First, the input images, originally 200×360 color images, were downsampled to 8×16 grayscale images. Second, the cGAN was initially trained using a deep convolutional GAN (DCGAN), but was then replaced by a smaller feed-forward neural network that emulates the image generation process of the DCGAN, in order to simplify analysis by the neural network verification tool. The detail architecture of the unified network is listed in Table S1. Also, two pairs of real images and the generated images using the network are shown in Fig. S1.

Table S1: Architectures of neural networks in autonomous aircraft taxiing system.

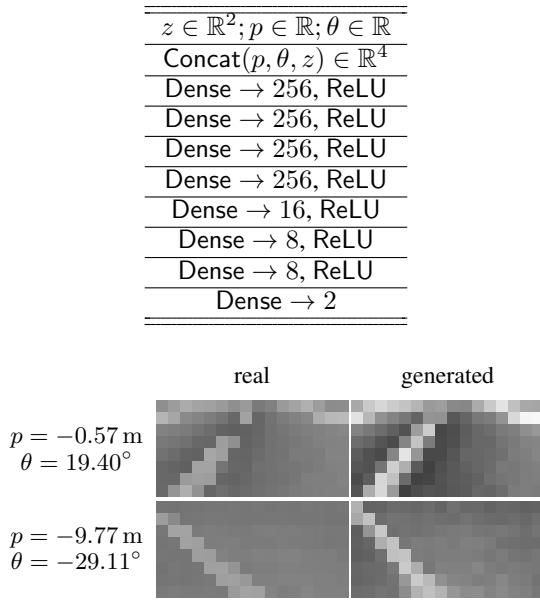


Figure S1: Real images and generated images for the taxiing system, the case study considered by the baseline method.

B.2 Linearization Technique for the Dynamics

To account for the nonlinear dynamics of aircraft taxiing system, we employ a conservative linearization technique based on Taylor expansion (Althoff, Stursberg, and Buss 2008). This technique approximates the nonlinear dynamics with a 1st order Taylor series and its 2nd order remainder:

$$x_{k+1}^i \in \underbrace{f^i(z_k^*) + \frac{\partial f^i(z)}{\partial z} \Big|_{z_k^*}}_{A_{(i,\cdot)}} (z_k - z_k^*) + L_i,$$

where $x^T = [p, \theta]^T$ is the state vector, $z^T = [p, \theta, \phi]^T$ combines the states and input, and z^* is the expansion point, typically located at the center of the star set. Specifically, $A = \begin{bmatrix} 1 & v\Delta t \cos \theta & 0 \\ 0 & 1 & \frac{v}{L}\Delta t \tan^2 \phi \end{bmatrix}$, and the Lagrange remainders for the two state variables can be separately overapproximated using interval arithmetic (Jaulin et al. 2001) as:

$$\begin{aligned} L_1 &\leq \frac{1}{2}v\Delta t[\min(-\sin \theta \cdot (\theta - \theta^*)^2), \\ &\quad \max(-\sin \theta \cdot (\theta - \theta^*)^2)], \\ L_2 &\leq \frac{v}{L}\Delta t[\min(\tan \phi \cdot (\tan^2 \phi + 1) \cdot (\phi - \phi^*)^2), \\ &\quad \max(\tan \phi \cdot (\tan^2 \phi + 1) \cdot (\phi - \phi^*)^2)]. \end{aligned}$$

B.3 Additional Experiments

The importance of our accuracy improvements becomes more apparent when parameters of the case study are altered.

To reduce verification time, we could consider a coarser grid with fewer cells. However, this results in more abstraction error. Another possible variant could consider a modified controller that converges more slowly. The convergence speed of the controller is important for verification, as it reduces the size of the reachable set and can compensate for overapproximation error in the analysis. We detail results on these variants in Fig. S2. In both scenarios, reachable sets computed by the baseline method extend beyond the taxiway, and P1 cannot be verified. The proposed methods effectively mitigate overapproximation, and successfully verify the safety of P1.

C Appendix for Advanced Emergency Braking System

C.1 Network Architecture and Generated Images

We detail the network architectures for both convolutional and transformer variants in Table S2 and compare the real and generated images for both network variants in Fig. S3.

Table S2: Architectures of NNs in braking system; ResBlock and SelfAttention blocks refer to Self-Attention GAN (Zhang et al. 2019).

(a) Generator (convolutional network)

$z \in \mathbb{R}^4 \sim N(0, I); c \in \mathbb{R}$
Concat(c, z) $\in \mathbb{R}^5$
Dense $\rightarrow 2 \times 2 \times 128$, BN
$(4 \times 4) \times 128$, stride = 2 ConvTranspose, BN, ReLU
$(4 \times 4) \times 64$, stride = 2 ConvTranspose, BN, ReLU
$(4 \times 4) \times 32$, stride = 2 ConvTranspose, BN, ReLU
$(3 \times 3) \times 1$, stride = 1 ConvTranspose, Tanh

(b) Controller (convolutional network)

Grayscale image $\hat{o} \in \mathbb{R}^{32 \times 32 \times 1}$
$(3 \times 3) \times 16$, stride = 2, padding = 1 Conv, ReLU
$(3 \times 3) \times 32$, stride = 2, padding = 1 Conv, BN, ReLU
$(3 \times 3) \times 64$, stride = 2, padding = 1 Conv, BN, ReLU
$(3 \times 3) \times 128$, stride = 2, padding = 1 Conv, BN, ReLU
Dense $\rightarrow 1$ ($\hat{d} \in \mathbb{R}^1$)
Concat(\hat{d}, v) $\in \mathbb{R}^2$
Dense $\rightarrow 400$, ReLU
Dense $\rightarrow 300$, ReLU
Dense $\rightarrow 1$, Clamp(0, 1)

C.2 Quantitative Results of Verifying the Braking System

Fig. 6 in the main text illustrates the set of states guaranteed to satisfy the safety property for different controllers in the braking system. Detailed quantitative results for this figure are provided in Table S3.

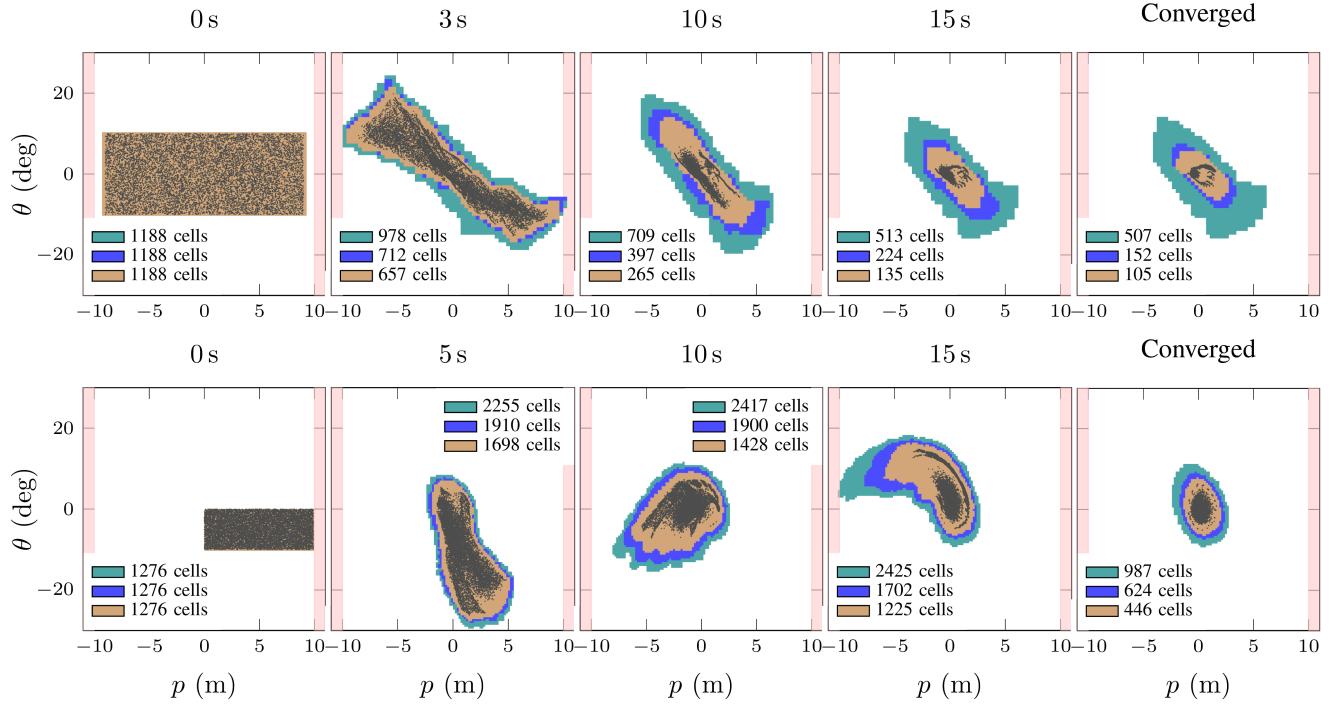
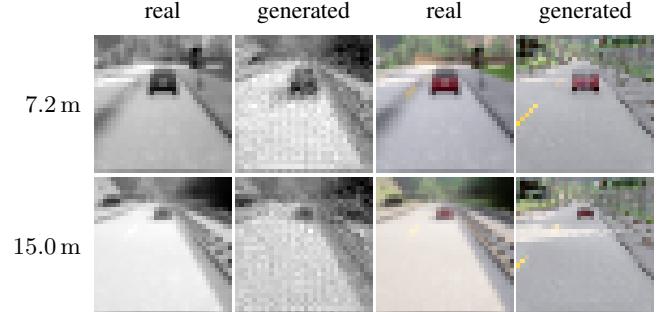


Figure S2: Reachable sets over time using three different methods in the taxiing system. The colors black, teal, blue, and brown are used to represent simulations, baseline, 1-step, and 2-step methods, respectively. The color pink denotes unsafe regions. Top: the height and width of the discretization cell are doubled, and the initial set is defined with $p \in [-9 \text{ m}, 9 \text{ m}]$ and $\theta \in [-10^\circ, 10^\circ]$. The baseline and 1-step methods fail to prove safety of the system at 3 s, whereas 2-step method succeeds. Bottom: the image-based controller is trained with less episodes, and the initial set is defined with $p \in [0 \text{ m}, 9.9 \text{ m}]$ and $\theta \in [-10^\circ, 0^\circ]$. The baseline method fails to prove safety of the system at 15 s, whereas 1-step and 2-step methods succeed.

(c) Generator (transformer network)
$z \in \mathbb{R}^4 \sim N(0, I); c \in \mathbb{R}$
Concat($c, z) \sim \mathbb{R}^5$
Dense $\rightarrow 4 \times 4 \times 256$
ResBlock up, 256
ResBlock up, 256
SelfAttention, 256
ResBlock up, 256
BN, ReLU, $(3 \times 3) \times 3$, stride = 1 Conv, Tanh



(d) Controller (transformer network)
RGB image $\hat{o} \in \mathbb{R}^{32 \times 32 \times 3}$
ResBlock down, 128
SelfAttention, 128
ResBlock down, 128
ResBlock, 128
ResBlock, 256, ReLU
Dense $\rightarrow 1$, Sigmoid, ($\hat{d} \in \mathbb{R}^1$)
Concat($\hat{d}, v) \in \mathbb{R}^2$
Dense $\rightarrow 400$, ReLU
Dense $\rightarrow 300$, ReLU
Dense $\rightarrow 1$, Clamp($0, 1$)

Figure S3: Real images and generated images for the braking system. The grayscale images are from the convolutional network; the color images are from the transformer network.

D Verification Times

We also provide the verification times for both the taxiing and braking systems in Table S4. For example, verifying the braking system with a 10 Hz controller using 1-step method requires approximately 125.14 GPU-hours (around 5 days). This time increases to 374.26 GPU-hours (around 15 days) for the 3-step method.

Additionally, as the system dimension exceeds two dimensions, the verification time could grow exponentially.

Table S3: Quantitative results of states identified as unsafe through simulations and states guaranteed to satisfy the safety property using 1-step, 2-step, and 3-step methods for different controllers in the braking system. Conv: Convolutional network variant; TF: Transformer network variant.

	Sim (Unsafe)	1-step	2-step	3-step
Conv (5Hz)	3463	0	384	2669
Conv (10Hz)	3521	390	2892	5739
Conv (20Hz)	3617	2859	5883	6035
TF (10Hz)	3517	390	2890	5707

While improving the efficiency of neural network verification tools is beneficial for the efficiency of closed-loop verification, it is crucial to design a more effective verification strategy.

Table S4: Verification times for all cells in the state graph. Note that these times are normalized to the times required by a single machine. These times are provided for reference, as experiments are conducted in parallel on multiple machines with different CPU and GPU configurations. The reported times are cumulative results across all machines.

	Taxiing system	Braking system	
		Convolutional	Transformer
1-step method	51.16 h	125.14 h	2630.88 h
2-step method	3954.77 h	168.67 h	3113.45 h
3-step method	N/A	374.26 h	3609.23 h