A Comprehensive Safety Analysis for Tracking Neural Controllers

Vladislav Nenchev

***Abstract*— This paper presents an offline safety analysis framework for neural controllers in model predictive control settings. Building on the insight that a (sensed) reference trajectory can be treated as a disturbance preview, our method constructs a non-adversarial controlled invariant set that both captures realistic operating conditions and includes the adversarial disturbance set. This formulation reduces safety checking to a collection of input–output constraints on the neural network. Leveraging a neural network verifier, our approach identifies counterexamples even when nominal performance matches that of a model predictive controller baseline. In a lane-keeping automated-driving case study, we falsify three neural controllers within minutes. We demonstrate that disturbance preview reduces false alarms in counterexample search and that the workflow scales to networks of practical size. Thus, the framework offers an effective rigorous safety certification of learned controllers in cyber-physical systems.**

I. Introduction

Artificial Intelligence (AI) integration has become increasingly popular in control systems, offering efficient approximation that can significantly reduce the memory footprint and online computation compared to traditional Model Predictive Control (MPC) implementations [1], [2]. In embedded and real-time applications – such as automated driving, robotics, aerospace, and emerging Edge AI platforms – Deep Neural Networks (DNNs) promise faster execution and easier deployment on resource-constrained hardware. However, their black-box nature raises concerns in safety-critical settings, where formal guarantees on closed-loop behavior are required. Empirical testing of relevant behaviors alone often fails to uncover rare but catastrophic failures in autonomous systems, as documented in recent studies of self-driving vehicles [3], [4], [5].

Robust control methods address safety by operating within the Robust Controlled Invariant Set (RCIS), which ensures constraint satisfaction under worst-case disturbances [6]. While theoretically sound, this approach is often overly conservative: restricting operation to the RCIS can exclude relevant scenarios and degrade performance, and it does not directly apply to learned controllers whose decision logic is not easily explainable. Moreover, existing DNN-verification methods typically check simple input-output properties on feed-forward networks [7], [8], without considering closed-loop behavior or handling time-varying references.

In this paper, we propose an offline safety analysis framework that bridges these gaps for tracking DNN controllers in an MPC context. Based on the insight that a reference-tracking MPC can be reformulated as an MPC with disturbance preview, the sensed reference trajectory can be treated as a known exogenous input. This allows to compute a non-adversarial Controlled Invariant Set (CIS) [9] that captures realistic operating conditions beyond the worst-case disturbance envelope, e.g., using fixed-point methods [10]. Embedding this CIS into the network's inputs reduces safety properties to a set of constraints on the DNN, which is checked using an Satisfiability Modulo Theory (SMT)-based verifier (e.g. [7]). The output is either a formal guarantee of safety or a concrete counter-example input demonstrating constraint violation. The main contributions are:

(i) Reference tracking in MPC is casted as a disturbance preview, enabling integration with invariant-set methods.
(ii) A CIS is computed that reflects realistic disturbances and contains the worst-case set to yield less conservative yet rigorous safety bounds.
(iii) The CIS is encoded as input–output constraints on the DNN and an off-the-shelf DNN verifier is used to automatically certify safety or produce counterexamples.
(iv) The framework is applied to DNN controllers for a Lane Keeping Assist System (LKAS). Despite matching nominal MPC performance, each controller is falsified in minutes.

Recent studies have highlighted the challenges of ensuring the safe behavior of DNN-controlled systems, particularly in the presence of disturbances [11]. For complex systems such as automated driving vehicles, traditional testing [12] is often accompanied by searching for specification counterexamples in simulation [13]. Formal verification may provide enhanced safety assurance compared to traditional testing. Various methods have been proposed for formally verifying DNNs, particularly for deep feed-forward [7] and convolutional neural networks [8], but these approaches typically focus on simple input-output properties. Some AI-based solutions can facilitate safe decision-making in uncertain robotic environments [14], but many have only been validated on small-scale problems [15] or are limited to specific neural network architectures [16]. Alternatives such as offline monitoring [17] have been proposed, but they may encounter challenges related to real-time performance and scalability. Recent works use the CIS for checking the safety of DNN-based automated driving controllers [18], [19], yet none combine disturbance-preview invariant sets with offline DNN safety analysis in an MPC setting for safety-critical embedded applications.

The remainder of the paper is organized as follows. Sec-

V. Nenchev is with the Department of Electrical and Computer Engineering, University of the Bundeswehr Munich, Werner-Heisenberg-Weg 39, 85579 Neubiberg, Germany. vladislav.nenchev@unibw.de

tion II formalizes the tracking control problem and introduces disturbance preview. Section III presents our non-adversarial invariant-set computation and the safety analysis framework. Section IV details the LKAS case study and discusses performance, followed by a conclusion in Section V.

Notation. An $m \times n$ zero matrix is denoted $0_{m,n}$. For $\alpha \in \mathbb{R}$ and $X \subset \mathbb{R}^n$, $\alpha X := \{\alpha x \mid x \in X\}$. $[0, N]$ denotes an integer set from 0 to N.

II. Problem statement

Consider a controller given as a DNN for a dynamical system with state $x_t \in X \subset \mathbb{R}^n$, control $u_t \subset U \in \mathbb{R}^m$, and a disturbance $d_t \in D \subset \mathbb{R}^l$. The compact disturbance set D contains all possible disturbances. Assume that predictions of the disturbances are available at each sampling time t, such as for an external signal previewed ahead by sensors. For instance, for the LKAS case study the disturbance is the orientation of the reference. Consequently, the input of the DNN comprises the current system state x_t and the disturbance predictions $d_{[0,N-1]|t} = [d_{0|t}^T, d_{1|t}^T, \ldots, d_{N-1|t}^T]^T$, where each $d_{\cdot|t} \in D$. Without loss of generality, the output of the DNN is the control u_t in this work. An extension for the case when the DNN output is a vector of controls or a state trajectory over the planning horizon is readily possible, as outlined in following remarks.

A. Deep neural network controller

The DNN is a composition of multiple layers of neurons, where each layer applies a linear transformation followed by a non-linear activation function. Each layer i performs $\tilde{z}^{(i)} = \sigma^{(i)}(W^{(i)} z^{(i)} + b^{(i)})$, where $\tilde{z}^{(i)}$ represents the output, $z^{(i)}$ is the input, $\sigma^{(i)}$ is the activation function, $W^{(i)}$ is the layer weight matrix, and $b^{(i)}$ is the layer bias vector. Assume that the activation functions are given as a Rectified Linear Unit (ReLU), i.e., $\sigma^{(i)}(y) = \max\{0, y\}$. Thus, given the input $\tilde{x}_t = [x_t^T, d_{0|t}^T, \ldots, d_{N-1|t}^T]^T$ the DNN controller with L layers is a mapping $g : \mathbb{R}^{n+N} \to \mathbb{R}^m$ with:

$$u = g(\tilde{x}_t) = W^{(L)} \sigma(\ldots \sigma(W^{(1)} \tilde{x}_t + b^{(1)}) + \ldots) + b^{(L)}. \tag{1}$$

The DNN has been trained by a suitable machine learning technique, such as imitation or reinforcement learning [20].

B. System model and safety objective

Consider the discrete-time system model:

$$x_{t+1} = Ax_t + Bu_t + Ed_t, \tag{2}$$

where A, B and E are real matrices of appropriate dimension. Although we focus on models of the form (2), the safety analysis framework proposed in this paper can be readily adapted to smooth nonlinear systems, as outlined in the discussion. Let $O = X \times U$ denote the operation set of admissible state-control pairs. Our goal is to certify that, under all admissible disturbances, the closed-loop operation remains in O. As the control u_t is limited, the safe operation set S will be a subset of O, i.e., $S \subset O$.

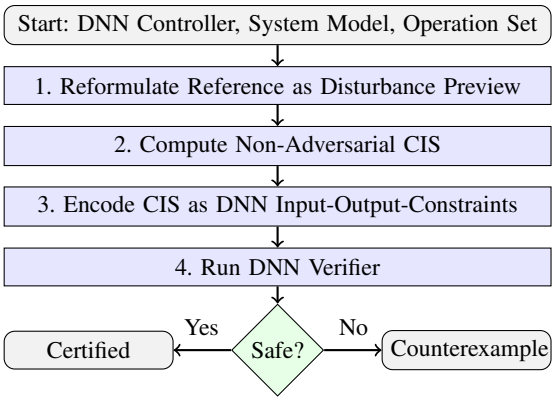


Fig. 1: The safety analysis framework.

C. Problem formulation

Problem 1: Given the DNN controller (1) and the system model (2), determine whether for any state $x_t \in X$ and disturbance predictions $d_{[0,N-1]|t}, d_{i|t} \in D$, the controller provides controls $u_t \in U$ that produce following states $x_{t+1} \in O$ with (2).

This problem cannot be solved directly, due to the following reasons. First, only one disturbance is contained in the model (2) at time t, but the DNN input contains a disturbance prediction vector $d_{[0,N-1]|t}$. Picking random values for disturbance predictions within D might not allow any safe following system state, as the control is bounded. Second, due to the presence of bounded controls and disturbances, it is hard to define S analytically. In practice, the system may be initialized or be pushed into a challenging state by a sporadic large disturbance, which may still allow safe operation over time, e.g., if the disturbance disappears in the following evolution of the system. Third, the safety property for the DNN controller must be represented in a form that allows rigorous and scalable automatic analysis.

III. Safety Analysis

This section presents the safety analysis framework for DNN controllers (Fig. 1). First, we augment the nominal plant model by treating the reference or disturbance forecast as an explicit preview input, yielding an auxiliary system. Second, we compute a non-adversarial CIS for this augmented model – the largest subset of the operation envelope from which the controller can guarantee safety under an adversarial perturbation while exploiting non-adversarial preview. Third, we derive a finite set of inequalities over the DNN inputs and outputs that enforces one-step invariance of the CIS. Finally, these constraints are encoded into an off-the-shelf DNN verifier to automatically certify safety or produce counterexamples.

A. Augmented System Model

The disturbance prediction can be integrated into the system (2) by additional states corresponding to the preview of the disturbance. Specifically, the following augmented

system is obtained:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + Ed_{0|t} \\ d_{0|t+1} &= d_{1|t} \\ &\vdots \\ d_{N-1|t+1} &= d_{N|t}, \end{aligned} \tag{3}$$

where $d_{N|t} \in D$ is a non-measurable disturbance. The operation set of (3) is denoted by $O_d \subseteq X \times U \times D^N$.

Remark 1: If the output of the DNN is a control vector over the planning horizon, additional states can be integrated into the system (3) for each control value.

B. Computing the safe set

As the augmented system (3) is subject to constrained controls and disturbances, its safe set S_d will be computed by a CIS, i.e., the set of states that the system can enter and remain in indefinitely under the influence of its dynamics, controls and disturbances. Let a disturbance model be a function that assigns a subset of D to each state $x \in X$. Most existing research on safety control focuses on the case, where the disturbance model assumes that the worst case disturbance acts always on the system. The corresponding maximal RCIS is typically a very small portion of the operation set O_d, such that an analysis with respect to it does not cover many practically relevant states. An example of such a state is an $x_t \in S_d$ located close to the boundary of S_d, but when almost no disturbance is acting the system can resume safe operation within S_d in the following steps.

To obtain the set of disturbance models to which a controller can possibly be robust to when the system is in state x_t, as inspired by [9], the disturbance is decomposed into two components as $\mathcal{D} = \{u^d + d \mid u^d \in (1-\alpha)D, d \in \alpha D, \alpha \in [0,1]\}$, treating u^d as an additional control. This set is sufficiently rich, as it encompasses uncountably many subsets of D that maintain the same shape as D and are scaled to different sizes and positioned at various locations in D. Thus, introduce a corresponding new state variable $\alpha_i \in [0,1]$ and a new control $u_i^d \in \mathbb{R}^l$ for each disturbance prediction, forming the vectors $\alpha = [\alpha_0, \dots, \alpha_{N-1}]^T$ and $u^d = [u_0^d, \dots, u_{N-1}^d]^T$. Then, the system (3) is extended to:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + E(u_{0|t}^d + d_{0|t}) \\ d_{0|t+1} &= u_{1|t}^d + d_{1|t} \\ &\vdots \\ d_{N-1|t+1} &= u_{N-1|t}^d + d_{N-1|t} \\ \alpha_{0|t+1} &= \alpha_{0,t} \\ &\vdots \\ \alpha_{N-1|t+1} &= \alpha_{N-1,t}. \end{aligned} \tag{4}$$

with corresponding operation set $O_{d\alpha} = X \times U \times D^N \times \alpha^N$. The maximal RCIS of (4) with respect to the operation set $O_{d\alpha}$ is denoted by $S_{\max,\alpha}$. When $\alpha = 1_N$, $S_{\max,1}$ corresponds to the maximal RCIS with respect to O_{d1} assuming the worst-case disturbance. When $\alpha = 0_N$, $S_{\max,0}$ denotes the maximal CIS of the undisturbed system. Note that all $S_{\max,\alpha}$ for $\alpha_i \in (0,1)$ are subsets of $S_{\max,0}$. Therefore, for any initial state x_0 not contained in $S_{\max,0}$, it is impossible to find a controller and a disturbance model such that the system is safe. Since both O, D and α are polytopes, $O_{d\alpha}$ is also a polytope, and the maximal RCIS $S_{\max,\alpha}$ can be approximated using standard iterative methods such as [21].

Remark 2: If the maximal RCIS $S_{\max,\alpha}$ cannot be computed exactly, for example when (4) is non-linear, any controlled invariant inner approximation of $S_{\max,\alpha}$ can be utilized for the safety analysis, although at the cost of increased conservativeness.

C. Safety analysis by model checking

Assuming that $S_{max,\alpha}$ is described by n_s inequalities $A_s\hat{x} \leq B_s$ with $\hat{x} = [x_t^T, d_{[0,N-1]|t}^T, \alpha^T]^T$, the DNN safety property is given by the following finite set of inequalities:

$$u_t \in U, \tag{5}$$

$$\begin{bmatrix} A_s & 0_{n_s,m} \end{bmatrix} \begin{bmatrix} \hat{x} \\ u_t \end{bmatrix} \leq B_s, \tag{6}$$

$$\begin{bmatrix} A_s & 0_{n_s,m} \end{bmatrix} \begin{bmatrix} A & 0_{n,2N} & B \\ & 0_{2N+m,2N+n+m} & \end{bmatrix} \begin{bmatrix} \hat{x} \\ u_t \end{bmatrix} \leq B_s. \tag{7}$$

This ensures that any control u_t provides robust safety guarantees when the system operates within the maximal RCIS $S_{\max,1}$, and that the DNN control u_t offers robustness guarantees within the extended CIS $S_{\max,0\leq\alpha<1}$ for smaller disturbances. Even if $S_{\max,1}$ is empty, the DNN may provide a control u_t that allows to remain in the extended CIS $S_{\max,0\leq\alpha<1}$. Thus, Problem 1 is reformulated as follows.

Problem 2: Given the DNN controller (1) and the safe set $S_{max,\alpha}$, check if for any state $x_t \in S_{max,\alpha}$, disturbance prediction $d_{[0,N-1]|t}, d_{i|t} \in S_{max,\alpha}$, and $\alpha_i \in S_{max,\alpha}$ that satisfy (5) and (6), the DNN-based controller (1) provides controls such that (7) holds.

The safety analysis is thus reduced to assigning DNN input values that satisfy all safety constraints simultaneously or confirming that no such assignment exists.

To verify the correctness of the DNN and its adherence to safety specifications (5)-(7), model checking is employed as a final step. Problem 2 can be formulated directly as a Mixed-Integer Linear Program (MILP). However, the runtime of the MILP-based approach increases exponentially with deeper networks. Some specialized DNN verifiers scale linearly with the number of layers [22]. Consequently, a DNN verifier is employed to assess the safety of the DNN. Along with the DNN, the verification query submitted to the verifier includes a property to be evaluated. The tool [7] is used to answer queries about a network's properties (5)-(7) by transforming these queries into constraint satisfaction problems. However, the property (5)-(7) to be checked over the inputs $\hat{x}$ and output u_t of the DNN has a higher dimension than the input $\tilde{x}$ of the DNN (1), as it contains α as an additional variable. Therefore, the input vector of (1) is extended without modifying the DNN behavior. This is accomplished by adding N additional zero columns to the right of the

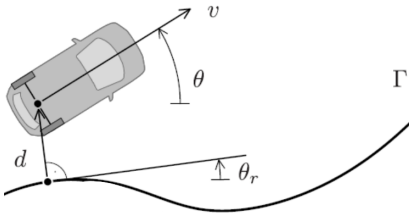


Fig. 2: Lateral vehicle kinematics [23].

weight matrix $W^{(1)}$ of the DNN (1) to account for α. The weights associated with the original inputs as well as the overall DNN behavior remain unchanged. At the same time, the DNN verifier is free to choose α values within admissible bounds to cover the range of possible disturbances.

Remark 3: If the output of the DNN is a control vector over the planning horizon, the DNN safety property is given by multiple instances of (5) and (7) for each control. If the output of the DNN is a state vector over the planning horizon, let $\tilde{x}_i = [x_i^T, 0_{1,N}, 0_{1,N}]^T$ for each output state. Then, the overall state for model checking is $\hat{x} = [x_t^T, d_{[0,N-1]|t}^T, \alpha^T, \tilde{x}_{t+1}, \ldots, \tilde{x}_{N-1}]^T$ and the DNN safety property is given by (6) with appropriate dimension and multiple instances of $A_s\tilde{x}_i \leq B_s$ for each output state. The combined case – where the outputs include both a vector of controls and states – can be addressed in a similar manner.

IV. Case Study

This section presents a case study that demonstrates the application and effectiveness of the proposed safety analysis for DNN controllers for automated driving. Consider an automated driving vehicle that has to keep its lane in a highway driving scenario (Figure 2). Given a reference curve Γ, which denotes the center of the lane, linearized relative kinematics are assumed. The vehicle's rear axle center is used as a reference point, d represents the normal signed distance between the reference curve and the center of the rear axis of the vehicle (known as the cross-track error), θ is the vehicle orientation, and the curvature κ is the control. The orientation θ_r is the normal distance between the reference curve and the vehicle's rear axle center position, serving as a disturbance to the model. The state contains θ and d, yielding

$$x_{t+1} = Ax_t + B\kappa_t + E\theta_{r,t}, \text{ with}$$
$$A = \begin{bmatrix} 1 & 0 \\ vt_s & 1 \end{bmatrix}, B = \begin{bmatrix} vt_s \\ \frac{1}{2}vt_s^2 \end{bmatrix}, E = \begin{bmatrix} 0 \\ -vt_s \end{bmatrix}, \tag{8}$$

for a constant longitudinal velocity of $v = 30\,\text{m/s}$ and sampling time $t_s = 0.2\,\text{s}$. The operation set is given as

$$O = \{|d| \leq 2\,\text{m} \wedge |\theta| \leq \pi/2\,\text{rad} \wedge |\kappa| \leq 0.2\,\text{m}^{-1}, |\theta_r| \leq \pi/10\,\text{rad}\}. \tag{9}$$

A. Data generation

An MPC is designed using the model (8) with an optimization horizon of $N = 5$ samples. Let t denote the current time step and $\tilde{t}$ the optimization horizon time. Given the initial state x_t and the disturbance prediction $d_{[0,N-1]|t} = [\theta_{r,0|t}^T, \theta_{r,1|t}^T, \ldots, \theta_{r,N-1|t}^T]^T$, a linear program is solved to find the maximal α^* such that $(x_t, d_{[0,N-1]|t}) \in S_{\max,\alpha}$. The corresponding slice $S_{\max,\alpha^*}$ is used as a terminal set for the states and P is an $n \times n$ matrix representing costs on the terminal states, obtained as the solution to the corresponding infinite discrete-time Linear Quadratic Regulator problem. With the weight of the control $w_r = 0.1$, the following Quadratic Program (QP) is formulated:

$$\begin{aligned} \min_{\kappa_{[\tilde{t},\tilde{t}+N-1]}} & \sum_{\tilde{t}=0}^{N-1} ((x_{\tilde{t}})^T x_{\tilde{t}} + w_r\kappa_{\tilde{t}}^2) + x_{\tilde{t}+N}^T P x_{\tilde{t}+N}, \\ \text{s.t. } & \forall \tilde{t} \in [0,N], (8), x_{\tilde{t}} \in O, \kappa_{\tilde{t}} \in O; \\ & x_{\tilde{t}} = x_t, (x_{\tilde{t}+N}, d_{[0,N-1]|t}) \in S_{\max,\alpha^*}. \end{aligned} \tag{10}$$

The QP problem, characterized by 288 constraints and 72 continuous optimization variables, is solved and applied in a receding horizon manner, i.e., at each time step only the first control action is applied, and then the process is repeated. The generated data contains 10,000 different MPC runs, each corresponding to a unique initial state and disturbance prediction vector, uniformly distributed within the bounded polyhedron of the invariant $S_{\max,\alpha}$. Note that while the maximal RCIS is utilized in the controller, it is only implicitly considered through the data for DNN training.

B. Training

As deep network architectures, which consist of multiple hidden layers, demonstrated superior performance in terms of training error and memory efficiency for some functions compared to shallower alternatives [1], the considered network architecture is composed of several fully connected layers with ReLU activation functions except for the output layer where a linear function is used. Various network architectures were empirically explored, including shallow and deep networks, to determine a suitable configuration. Finally, three DNNs with different architectures were designed. DNN 1 is a shallow network featuring a single hidden layer with 40 neurons, while DNN 2 has two hidden layers each with 20 neurons. DNN 3 is a deeper network comprising 4 hidden layers, each containing 10 neurons. All DNNs have the same number of inputs $n + N$ and outputs $l = 1$. For the training of the DNN, the Adam optimizer is employed. The training process is conducted using 90% of the generated data as training data, while the remaining 10% is reserved as evaluation data. This division allows for the conventional assessment of the performance of the trained model and ensures that generalization to unseen data is achieved. Figure 3 illustrates the training error for the three network architectures. The evaluation error closely resembles the training error, indicating that overfitting is not present; thus, it is omitted here for conciseness. Although all DNNs have the same total number of neurons, DNN 3 achieves a much lower training error. The performance and robustness of this controller will be assessed in the following.

In this section, a conventional performance evaluation of the DNN controller is undertaken. For that, a disturbance trajectory is concatenated from samples of disturbance vectors $d_{[0,N-1]|t}$ from the evaluation data set. Figure 4 shows a

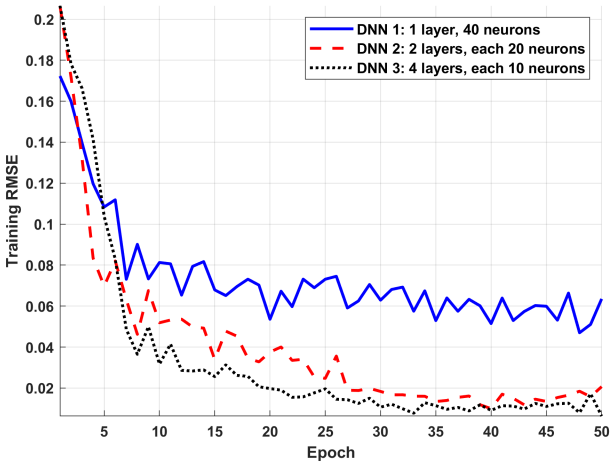
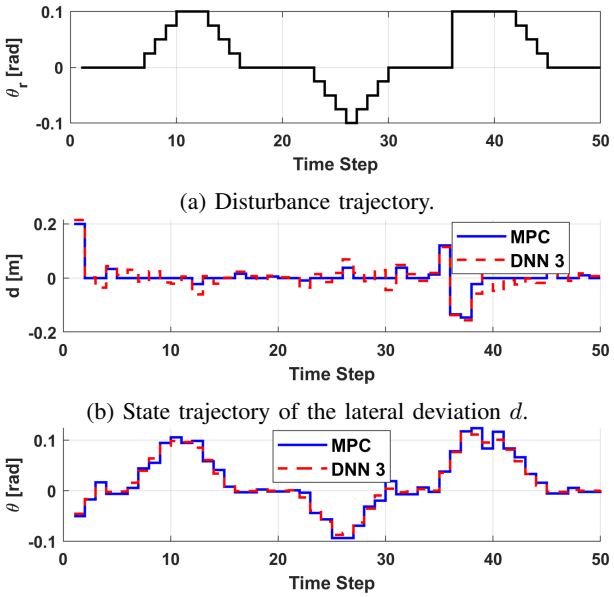


Fig. 3: Training error for different network architectures.



(a) Disturbance trajectory.

(b) State trajectory of the lateral deviation d.

(c) State trajectory of the angle θ.

Fig. 4: Performance of the controllers with concatenated disturbance trajectories from the evaluation set.

comparison between the MPC controller that solves the QP (10) online using the solver LCP and the DNN controller DNN 3 that solely evaluates the DNN trained in the previous subsection. The same initial conditions and disturbance trajectories are utilized for both solutions. Even for unseen scenarios, a very similar performance is achieved by the deep learning-based controller compared to the MPC that solves QPs to global optimality at each sampling interval.

C. Safety analysis results

The CIS is computed as described in Sec. III using a fixed-point iteration approach [10]. Figure 5 shows invariants for different values of α and $u^d = 0$. Geometrically, for $0 \leq \alpha < 1$, $u^d \neq 0, u^d \in (1-\alpha)D$ shifts the corresponding safe set along the y-axis according to the dynamics (8). The SMT-based tool [7] is used as a DNN verifier to check the safety

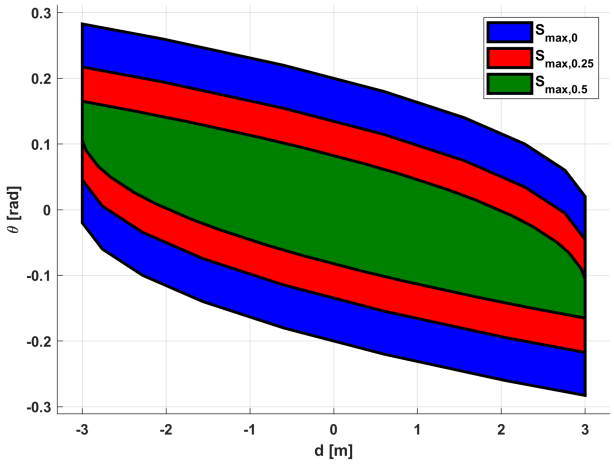


Fig. 5: Safe sets for varying α (with $u^d = 0$); contours show the boundary of $S_{\max,\alpha}$.

TABLE I: Verification/falsification times of the DNNs with different number of weights n_{weights} (including biases) and upper bound disturbance parameters $\alpha_{\max}$.

DNN №	Neuron config	$\alpha_{\max}$	n_{weights}	falsification time $[min]$	verification time $[min]$
1	40	1	361	3.0	–
2	20-20	1	601	5.1	–
3	10-10-10-10	1	421	6.1	–
3	10-10-10-10	0.5	421	5.3	–
3	10-10-10-10	0.25	421	–	4.2

properties of the DNNs, trained as described in Sec. IV-B on a standard workstation with an Intel Core i7-11850H CPU with 64 GB DDR4 RAM.

As shown in Sec. IV-B, the performance of the baseline MPC and the DNN controllers match for the test data set. The MPC tackles all scenarios safely, as both the RCIS and the disturbance prediction are explicitly considered in the problem formulation. However, when the safety analysis is applied to the DNNs, falsifying scenarios are identified within minutes. A particular emphasis is put on DNN 3, which exhibited the best performance among the trained DNNs. As the analysis stops when a falsifying DNN input is discovered, in some instances the upper bound $\alpha_{\max}$ of α was reduced to values below 1 to focus not only on the worst-case disturbances. The results of the analysis are summarized in Table I. DNN 3 provides safe controls only for small disturbances. Even for $\alpha_{\max} = 0.5$, which captures moderate disturbances, a falsifying control is found.

D. Discussion

The results demonstrate that our offline safety analysis framework is an important complement to standard performance evaluations for neural controllers. Although the performance of the studied DNNs matched MPC baselines on nominal test sets, our method uncovered safety violations highlighting the danger of relying only on empirical benchmarks. Unlike classical RCIS, which captures only states

safe under all possible disturbances, our non-adversarial CIS also contains states that are safe under the admissible disturbance previews, thereby covering a richer set of real-world scenarios. By systematically exploring the non-adversarial disturbance space instead of solely worst-case scenarios, we achieve a comprehensive analysis of operational safety. This helps us to avoid both false negatives, where hazards within the set S but outside the worst-case subset remain undetected, and false positives, such as states that are part of the operational set O but lie outside the safe set S.

Although formulated for linear models, the framework extends to smooth nonlinear systems by bounding nonlinearities within the disturbance set D. This can be achieved by partitioning the state space into local regions, linearizing the dynamics on each region, and bounding the linearization error as an additional disturbance. Computing local CISs and combining them yields a global safe set, which can be used in the DNN verifier exactly as in the linear case.

The proposed approach is conditional on the fidelity of the underlying model: if the dynamics deviate significantly from the true plant, safety certificates and counterexamples may be invalid. Moreover, computing high-dimensional CISs and solving large SMT queries can be computationally intensive. Empirical benchmarking against verification alternatives based on MILP or abstract interpretation will further quantify the trade-off between runtime and conservatism. Combining offline certificates with runtime monitors may help to account for model-plant mismatches.

Beyond automated driving, our methodology applies to safety-critical cyber-physical systems with predictive control and learned components – ranging from aerospace navigation to industrial robotics and energy-grid management. By unifying invariant-set theory with state-of-the-art DNN verification, the framework equips practitioners with a rigorous and practical tool for certifying neural controllers.

V. Conclusions

This paper proposed an offline safety analysis framework for neural controllers in an MPC setting by reformulating reference tracking as disturbance preview and computing a non-adversarial controlled invariant set. Embedding this safe set as constraints on the network's inputs and outputs allows using off-the-shelf verifiers to deliver guarantees or counterexamples. In a lane-keeping case study, our pipeline falsified learned controllers in minutes, despite their nominal MPC-level performance, while reducing false alarms compared to classical robust invariants. By combining invariant-set theory, disturbance preview, and DNN verification, our framework helps to deliver certifiable, explainable, and edge-deployable neural controllers for safety-critical cyber-physical systems.

References

[1] I. Safran and O. Shamir, "Depth-width tradeoffs in approximating natural functions with neural networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2979–2987, PMLR, 06–11 Aug 2017.

[2] B. Karg and S. Lucia, "Deep learning-based embedded mixed-integer model predictive control," in *2018 European Control Conference (ECC)*, pp. 2075–2080, 2018.

[3] N. Rajabli, F. Flammini, R. Nardone, and V. Vittorini, "Software verification and validation of safe autonomous cars: A systematic literature review," *IEEE Access*, vol. 9, pp. 4797–4819, 2021.

[4] J. Zhang and J. Li, "Testing and verification of neural-network-based safety-critical control software: A systematic literature review," *Information and Software Technology*, vol. 123, p. 106296, 2020.

[5] J. Chae, S. Lee, J. Jang, S. Hong, and K.-J. Park, "A survey and perspective on industrial cyber-physical systems (icps): From icps to ai-augmented icps," *IEEE Transactions on Industrial Cyber-Physical Systems*, vol. 1, pp. 257–272, 2023.

[6] X. Tan, W. S. Cortez, and D. V. Dimarogonas, "High-order barrier functions: Robustness, safety, and performance-critical control," *IEEE Trans. on Automatic Control*, vol. 67, no. 6, pp. 3021–3028, 2022.

[7] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett, "The marabou framework for verification and analysis of deep neural networks," in *Computer Aided Verification* (I. Dillig and S. Tasiran, eds.), (Cham), pp. 443–452, Springer, 2019.

[8] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "Ai2: Safety and robustness certification of neural networks with abstract interpretation," in *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2018.

[9] Z. Liu, H. Chen, Y. Gao, and N. Ozay, "Opportunistic safety outside the maximal controlled invariant set," *IEEE Control Systems Letters*, vol. 7, pp. 3992–3997, 2023.

[10] M. Kvasnica, B. Takács, J. Holaza, and D. Ingole, "Reachability analysis and control synthesis for uncertain linear systems in mpt," *IFAC-PapersOnLine*, vol. 48, no. 14, pp. 302–307, 2015. 8th IFAC Symposium on Robust Control Design ROCOND 2015.

[11] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[12] X. Zhou, X. Gou, T. Huang, and S. Yang, "Review on testing of cyber physical systems: Methods and testbeds," *IEEE Access*, vol. 6, pp. 52179–52194, 2018.

[13] A. Favrin, V. Nenchev, and A. Cenedese, "Learning to falsify automated driving vehicles with prior knowledge," *IFAC-PapersOnLine*, 2020. IFAC World Congress 2020 (IFAC'2020), Berlin.

[14] P. L. Donti, M. Roderick, M. Fazlyab, and Z. Kolter, "Enforcing robust control guarantees within neural network policies," in *Int. Conf. on Learning Representations (ICLR)*, 2021.

[15] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. Volume 5, 2022, pp. 411–444, 2022.

[16] V. Nenchev, "Layer-stabilizing deep learning," *IFAC-PapersOnLine*, vol. 52, no. 29, pp. 286 – 291, 2019. 13th IFAC Workshop on Adaptive and Learning Control Systems (ALCOS).

[17] V. Nenchev, "Model checking embedded adaptive cruise controllers," *Robotics and Autonomous Systems*, vol. 167, p. 104488, 2023.

[18] V. Nenchev, C. Imrie, S. Gerasimou, and R. Calinescu, "Code-level safety verification for automated driving: A case study," in *Formal Methods (FM'24)*, vol. 14934 of *Lecture Notes in Computer Science (LNCS)*, p. 356–372, Springer, 2024.

[19] V. Nenchev, C. Imrie, S. Gerasimou, and R. Calinescu, "Compositional code-level safety verification for automated driving controllers," *Journal of Systems and Software*, 2025.

[20] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[21] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada, "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1294–1307, 2016.

[22] R. Ivanov, T. J. Carpenter, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verifying the safety of autonomous systems with neural network controllers," *ACM Trans. Embed. Comput. Syst.*, vol. 20, dec 2020.

[23] H.-J. Wu, V. Nenchev, and C. Rathgeber, "Automatic parameter tuning of self-driving vehicles," in *2024 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 555–560, 2024.