Failing with Grace: Learning Neural Network Controllers that are Boundedly Unsafe

Panagiotis Vlantis PANAGIOTIS.VLANTIS@DUKE.EDU
Duke University

Leila J. Bridgeman LEILA.BRIDGEMAN@DUKE.EDU
Duke University

Michael M. Zavlanos MICHAEL.ZAVLANOS@DUKE.EDU
Duke University

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

This work considers the problem of learning a feed-forward neural network controller to safely steer an arbitrarily shaped planar robot in a compact, obstacle-occluded workspace. When training neural network controllers, existing closed-loop safety assurances impose stringent data density requirements close to the boundary of the safe state space, which are hard to satisfy in practice. We propose an approach that lifts these strong assumptions and instead admits *graceful* safety violations, i.e., of a bounded, spatially controlled magnitude. The method employs reachability analysis techniques to include safety constraints in the training process. The method can simultaneously learn a safe vector field for the closed-loop system and provide proven numerical worst-case bounds on safety violations over the whole configuration space, defined by the overlap between an over-approximation of the closed-loop system's forward reachable set and the set of unsafe states.

Keywords: Safe learning, neural network control, reachability analysis.

1. Introduction

Recent progress in machine learning has furnished a new family of neural network controllers for robot systems that significantly simplify the design process. As these controllers are adopted in real-world systems, the ability to train neural networks with safety considerations becomes necessary.

The design of data-driven controllers that result in safe closed-loop systems has typically relied on methods that couple state-of-the-art machine learning algorithms with control Perkins and Barto (2003); Geibel and Wysotzki (2005). A popular approach employs control barrier functions to appropriately constrain the control inputs so that a specified safe subset of the state space remains invariant during execution and learning Li and Belta (2019); Ohnishi et al. (2019); Cheng et al. (2019). However, designing appropriate barrier functions for robotic systems operating in complex environments is generally hard. Additionally, conflicts between the reference control laws and the barrier certificates may introduce unwanted equilibria to the closed-loop system. While Robey et al. (2020); Lindemann et al. (2020) proposed a method to learn control barrier functions from expert demonstrations, this method requires dense enough sampled data and Liptchitz constants of the system's dynamics and corresponding neural network controller that are hard to obtain in practice.

Compared to the control barrier function methods discussed above that can usually only ensure invariance of a conservative subset of the set of safe states, backwards reachability methods can instead compute the exact set of safe states which, similar to control barrier function methods, can then be rendered invariant by an appropriate design of controllers that take over when the system approaches the boundary of that safe set Bajcsy et al. (2019); Fisac et al. (2018); Li et al. (2020). A common point in these methods is that they generally apply reachability analysis on the open loop dynamics without considering the specific structure of the reference neural network controller, owning to its complexity. Reachability analysis of neural networks is an actively studied topic and recent solutions have been proposed for verification Katz et al. (2017); Ruan et al. (2018); Dutta et al. (2017) and robust training Zhang et al. (2018, 2019) alike. These methods have been successfully adapted for estimating the forward reachable set of dynamical systems in feedback interconnection with feed-forward neural network controllers Dutta and Sankaranarayanan (2019); Huang et al. (2019); Hu et al. (2020); Xiang et al. (2018). Although these methods provide accurate over-approximations of the reachable set of neural network controllers, they only address the verification problem of already trained controllers and do not consider safety specifications.

In this paper, we propose a framework for learning safe neural network controllers that relies on reachability analysis techniques to encapsulate safety constraints in the training process. The unique aspects of this framework are that it (i) provides proven numerical worst-case bounds on safety violations over the whole configuration space, defined by the overlap between the over-approximation of the forward reachable set of the closed-loop system and the set of unsafe states, and (ii) controls the tradeoff between computational complexity and tightness of these bounds. Compared to the methods in Cheng et al. (2019); Robey et al. (2020); Bajcsy et al. (2019) that employ control barrier functions or Hamilton-Jacobi reachability to design fail-safe projection operators or supervisory control policies, respectively, that can be wrapped around pre-trained nominal neural network controllers that are possibly unsafe due to, e.g., insufficient data during training, here we directly train neural network controllers with safety specifications in mind. On the other hand, unlike the methods in Li et al. (2020); Ohnishi et al. (2019); Cheng et al. (2019) that also directly train safe neural network controllers using sufficiently many safe-by-design training samples, here safety of the closed-loop system does not depend on data points, but on safety violation bounds defined over the whole configuration space that enter explicitly the loss function as penalty terms during training. As a result, when our method fails to guarantee safety of the closed-loop system in the whole configuration space, it does so with *grace* by also providing safety violation bounds whose size can be controlled. Such guarantees on the closed-loop performance cannot be obtained using the methods in Li et al. (2020); Ohnishi et al. (2019); Cheng et al. (2019) that depend on the density of sampling.

Perhaps most closely related to this work is the method in Sun and Shoukry (2021), which also trains provably safe neural network controllers for robot navigation. Compared to Sun and Shoukry (2021), the proposed method is not restricted to ReLU neural networks, can accommodate any class of strictly increasing continuous activation functions, applies to non-point robots, and returns smooth trajectories that respect the robot dynamics. Finally, feasibility of the control synthesis problem in Sun and Shoukry (2021) strongly relies on the availability of sufficient data needed to train neural network weights that belong to the regions found to be safe. Our proposed method removes such strong assumptions on training data that are costly in robotics applications and instead allows for graceful safety violations, of a bounded magnitude that can be spatially controlled.

Notation: We will refer to an interval on the set of real numbers $\mathbb{R}$ as a simple interval, whereas Cartesian products of simple intervals will be referred to as a composite. An n-dimensional interval is a composite interval of the form $[x_{l,1}, x_{u,1}] \times [x_{l,2}, x_{u,2}] \times \ldots \times [x_{l,n}, x_{u,n}]$. Given a compact set $\mathcal{S} \in \mathbb{R}^n$ and a vector $\delta = [\delta_1, \delta_2, \ldots, \delta_n] \in \mathbb{R}^n$, $\text{vol}_\delta(\mathcal{S})$ denotes the volume of $\mathcal{S}$ after scaling the i-th dimension by the δ_i, i.e., $\mathrm{vol}_\delta(\mathcal{S}) = \prod_{i=1}^{n} \delta_i \cdot (x_{u,i} - x_{l,i})$. For brevity, $\mathrm{vol}(\mathcal{S})$ denotes $\mathrm{vol}_\delta(\mathcal{S})$ when $\delta = [1, 1, \ldots, 1]$. Let $\mathrm{B}_r \subset \mathbb{R}^2$ be the ball of radius $r \geq 0$ centered at the origin.

2. Problem Formulation

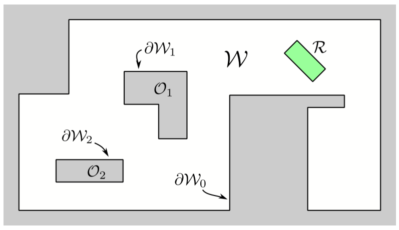


Figure 1: Robot, $\mathcal{R}$, operating in a workspace, $\mathcal{W}$, with two inner obstacles.

We consider a polygonal shaped robot $\mathcal{R}$ operating in a compact, static workspace $\mathcal{W} \subset \mathbb{R}^2$ defined by an outer boundary $\partial\mathcal{W}_0$ and $N_\mathcal{O}$ inner boundaries $\partial\mathcal{W}_i, i \in \mathfrak{I}_\mathcal{O} \triangleq \{1, 2, \ldots, N_\mathcal{O}\}$, corresponding to a set of disjoint, fixed inner obstacles $\mathcal{O}_i$, as seen in Figure 1. We assume that the boundary $\partial\mathcal{R}$ of the robot's body and the boundaries $\partial\mathcal{W}_0, \partial\mathcal{W}_i, i \in \mathfrak{I}_\mathcal{O}$ of the workspace are polygonal Jordan curves. Let $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{R}$ be coordinate frames embedded in the workspace and on the robot, respectively, and let $p = [x, y, \theta]^T$ denote the configuration of the robot on the plane, specifying the relative position $[x, y]^T \in \mathbb{R}^2$ and the orientation $\hat{n}(\theta) = [\cos\theta, \sin\theta]^T$, $\theta \in [0, 2\pi)$, of $\mathcal{F}_\mathcal{R}$ with respect to $\mathcal{F}_\mathcal{W}$. We assume that the robot is able to translate and rotate subject to the following discrete-time non-linear dynamics

$$z_{k+1} = f(z_k, u_k), \tag{1}$$

where $z = \left[p^T, q^T\right]^T \in \mathcal{Z} \subset \mathbb{R}^n$, and $u \in \mathbb{R}^m$ denote the robot's state and control input, respectively, and q denotes miscellaneous robot states, e.g., linear and angular velocities, accelerations, etc. We define $\mathcal{Z} \triangleq \mathcal{Z}_{[p]} \times \mathcal{Z}_{[q]}$, where $\mathcal{Z}_{[p]} = \mathcal{W} \times \mathbb{S}^1$, and $\mathcal{Z}_{[q]} \subset \mathbb{R}^{n-3}$ is a $(n-3)$-dimensional interval of miscellaneous safe robot states, e.g., allowed bounds on the robot's velocities. The non-linear function $f : \mathbb{R}^{n+m} \mapsto \mathbb{R}^n$ is assumed known and continuously differentiable. Furthermore, we assume that a function F is known which maps composite intervals $\mathcal{A}$ and $\mathcal{B}$ of $\mathcal{Z}$ and $\mathbb{R}^m$, respectively, to composite intervals of $\mathcal{Z}$ such that

$$f(z, u) \in F(\mathcal{A}, \mathcal{B}), \quad \forall z, u \in \mathcal{A} \times \mathcal{B}. \tag{2}$$

In order to steer the robot to a desired safe state $z^\star$, we equip the robot with a feed-forward multi-layer neural network controller $\phi : \mathbb{R}^n \mapsto \mathbb{R}^m$ that consists of N_ϕ fully connected layers, i.e.,

$$u = \phi(z) = \phi_{N_\phi}\big(\phi_{N_\phi - 1}(\ldots \phi_1(z) \ldots)\big), \tag{3}$$

$$\phi_i(x) = h_i(w_i \cdot x + b_i), \quad \forall i \in \mathfrak{I}_{N_\phi}, \tag{4}$$

where w_i, b_i, h_i denote the weight matrix, bias vector and activation function of the i-th layer, for all $i \in \mathfrak{I}_{N_\phi}$. Specifically, the i-th layer of ϕ consists of n_i neurons, i.e., $w_i \in \mathbb{R}^{n_i \times n_{i-1}}$, $b_i \in \mathbb{R}^{n_i \times 1}$, $h_i \in \mathbb{R}^{n_i \times 1}$, for all $i \in \mathfrak{I}_{N_\phi}$ where $n_0 = n$ and $n_{N_\phi} = m$.

To identify $w_i, b_i, i \in \mathfrak{I}_{N_\phi}$ that allow the controller ϕ to steer the robot to the desired state $z^\star$, we assume that we are given a set $\mathcal{D}$ consisting of points $(z_i, u_i) \in \mathcal{Z} \times \mathbb{R}^m$ sampled from robot trajectories in the set of safe robot states beginning at different random states $z \in \mathcal{Z}$ and terminating at $z^\star$. This sampling need not cover all possible states and actions. A state $z = \left[p^T, q^T\right]^T$ is said to be safe if $q \in \mathcal{Z}_{[q]}$ and the robot is entirely in the workspace at the corresponding configuration, i.e., $\mathcal{R}(p) \subset \mathcal{W}$, where $\mathcal{R}(p)$ denotes the robot's footprint which is a set of states that must be in the workspace to maintain safety when the robot is placed at $[x, y]^T$ with orientation θ. For a large enough parameter space of the network, ϕ, and dataset, $\mathcal{D}$, parameters w, b can be typically found by solving

$$\underset{w,b}{\text{minimize}}\ J(w,b) \tag{5}$$

$$J(w,b) = \lambda_B \cdot \sum\nolimits_{(z,u)\in\mathcal{D}} \left(u - \phi(z; w, b)\right)^2 + \lambda_R \cdot r(w,b)\,. \tag{6}$$

In the loss function (6), $w = \left[w_1, w_2, \ldots, w_{N_\phi}\right]$, $b = \left[b_1, b_2, \ldots, b_{N_\phi}\right]$, $r(\cdot)$ is a regularization term and λ_B, λ_R are positive constants. We remark that the controller obtained by solving (5)-(6) is expected to be safe only around the trajectories in the training dataset $\mathcal{D}$. This behavior is generally not desirable. Instead, it is desired that the robot dynamics (1) under the control law (3) ensure that the safe set of states remains invariant. Therefore, in this paper we consider the following problem.

Problem 1 *Given a static workspace $\mathcal{W}$, a polygonal robot $\mathcal{R}$ subject to dynamics f, a safe set of miscellaneous robot states $\mathcal{Z}_{[q]}$, and a dataset $\mathcal{D}$, train a neural network controller ϕ so that the closed-loop trajectories fit the data in the set $\mathcal{D}$ and the safe set, $\mathcal{Z}_s \triangleq \{(p,q) \in \mathcal{Z} | \mathcal{R}(p) \subset \mathcal{W}\}$, either remains invariant or possible safety violations are explicitly bounded.*

3. Methodology

In order to address Problem 1, we first employ standard learning methods to solve the optimization problem (5) and obtain initial values for the parameters w and b. As discussed before, the controller ϕ obtained at this stage is expected to be safe only around the points in $\mathcal{D}$, assuming that they have been sampled from safe trajectories and the network fits the dataset adequately. Next, we employ the subdivision method presented in subsection 3.1 to obtain a partition $\mathcal{P}$ of the safe space into cells that provide a tight over-approximation $\overline{\mathcal{Z}}_s$ of the robot's safe state space $\mathcal{Z}_s$. Using the over-approximation of the safe set $\mathcal{Z}_s$ as a set of initial robot states, in subsection 3.2, we compute an over-approximation $\overline{\mathcal{Z}}_c$ of the forward reachable set $\mathcal{Z}_c$ of the closed loop system under the neural network controller ϕ. Since the accuracy of the over-approximation $\overline{\mathcal{Z}}_c$ depends on the partition $\mathcal{P}$ of the over-approximation of the safe set $\overline{\mathcal{Z}}_s$, we also propose a method to refine the partition in order to improve the accuracy of the over-approximation $\overline{\mathcal{Z}}_c$. Finally, in subsection 3.3, we use the overlap between the over-approximation of the forward reachable set $\overline{\mathcal{Z}}_c$ and the set of unsafe states to design penalty terms in problem (5) that explicitly capture safety specifications. As the parameters w, b of the neural network ϕ get updated, so does the shape of the forward reachable set $\overline{\mathcal{Z}}_c$, which implies that the initial partition $\mathcal{P}$ of the over-approximation of the safe set $\overline{\mathcal{Z}}_s$ does no longer accurately approximate $\mathcal{Z}_c$. For this reason, we repeat the partition refinement and training steps proposed in subsection 3.2 and subsection 3.3 for a sufficiently large number of epochs N_{epochs}. The procedure described above is outlined in Algorithm 1.

3.1. Over-Approximation of the Safe State Space

To obtain a tight over-approximation $\overline{\mathcal{Z}}_s$ of the robot's safe state space, that will be used to compute the closed-loop system's forward reachable set and its overlap with the unsafe state space, we adaptively partition the safe state space $\mathcal{Z}_s$ into cells using the adaptive subdivision method proposed in Zhu and Latombe (1991). Specifically, we start with a composite interval enclosing the set of safe states $\mathcal{Z}_s$. Since numerous well-established methods exist to select cell coverings, we

Algorithm 1: Safety-aware controller design.

```
function $\{w,\ b\}$ =TRAIN($\epsilon_w,\ \epsilon_{V_p},\ \epsilon_{V_q},\ N_{\text{epochs}}$)
    $w, b \leftarrow$ Solve problem (5);
    $\mathcal{P} \leftarrow$BUILDPRTN($\mathcal{W}, \mathcal{R}, \epsilon_w$) ;                                  // Use Algorithm 2
    for $i$ in $1, 2, \ldots, N_{\text{epochs}}$ do
        $\mathcal{P} \leftarrow$REFINEPRTN($\mathcal{P},\ \epsilon_w,\ \epsilon_{V_p},\ \epsilon_{V_q};\ \mathcal{W},\ \mathcal{R},\ \mathcal{Z}_{[q]},\ f$);     // Use Algorithm 3
        $w, b \leftarrow$ Solve problem (8);                                               // Update $w, b$
    end
end
```

omit a detailed discussion of the construction. Then, we recursively subdivide this interval into subcells based on whether appropriately constructed under- and over-approximations of the robot's footprint intersect with the workspace's boundary. The key idea is that cells for which the under-approximation (resp. over-approximation) of the robot's footprint overlaps (resp. does not overlap) with the complement of $\mathcal{W}$ contain only unsafe (resp. safe) states and subdividing them any further will not improve the accuracy of the partition whereas cells which contain both safe and unsafe states should be further subdivided into subcells as they reside closer to the boundary of $\mathcal{Z}_s$. This procedure is repeated until cells that constitute the partition of safe state space either contain only safe states or intersect with the boundary of $\mathcal{Z}_s$ and their size is below a user-specified threshold.

To begin, recall that the set $\mathcal{Z}_s$ is defined as $\mathcal{Z}_{[p],s} \times \mathcal{Z}_{[q]}$ where $\mathcal{Z}_{[q]}$ is a composite interval and $\mathcal{Z}_{[p],s}$ is defined as the largest subset of $\mathcal{Z}_{[p]}$ such that $\mathcal{R}(z) \subseteq \mathcal{W}$ for all $z \in \mathcal{Z}_{[p],s}$. Thus, in order to compute $\overline{\mathcal{Z}}_s$, we need to find a valid over-approximation $\overline{\mathcal{Z}}_{[p],s}$ of $\mathcal{Z}_{[p],s}$. Additionally, we require that the over-approximation $\overline{\mathcal{Z}}_s$ is defined by the union of a finite number of composite intervals, i.e., cells. This construction is necessary to obtain the forward reachable set of the closed loop system using the method proposed in subsection 3.2. We now consider a composite interval $\mathcal{C}$ in the robot's state space $\mathcal{Z}$ which we shall refer to as a state space cell. Each state space cell $\mathcal{C}$ can be written as $\mathcal{C}_{[p]} \times \mathcal{C}_{[q]}$, where $\mathcal{C}_{[p]} \in \mathcal{Z}_{[p]}$ denotes a configuration space cell, i.e., a set of robot's positions and orientations. We notice that if $\mathcal{R}(p) \subseteq \mathcal{W}$ for all $p \in \mathcal{C}_{[p]}$, then the cell $\mathcal{C}_{[p]}$ consists entirely of safe robot configurations and thus must lie entirely inside $\mathcal{Z}_{[p],s}$. On the contrary, if $\mathcal{R}(p) \cap \mathcal{W} \neq \mathcal{R}(p)$ for all $p \in \mathcal{C}_{[p]}$, then the cell $\mathcal{C}_{[p]}$ consists entirely of unsafe robot configurations and thus must lie outside $\mathcal{Z}_{[p],s}$. Since checking the above conditions to classify cells as safe or unsafe is not easy due to the complex shape of $\mathcal{Z}_s$ and the robot, we instead employ for this purpose over- and under-approximations of the robot's footprint, $\overline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right)$ and $\underline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right)$, respectively, associated with the configuration space cell $\mathcal{C}_{[p]}$ (see Figure 3 in Vlantis et al. (2022)) that satisfy

$$\overline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right) \supseteq \bigcup_{p \in \mathcal{C}_{[p]}} \mathcal{R}(p), \text{ and } \underline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right) \subseteq \bigcap_{p \in \mathcal{C}_{[p]}} \mathcal{R}(p). \tag{7}$$

We remark that such over- and under-approximations of the robot's footprint can be easily computed for box shaped cells $\mathcal{C}_{[p]}$ using techniques such as the *Swept Area Method* Zhu and Latombe (1991). Additionally, we notice that a cell $\mathcal{C}_{[p]}$ for which $\overline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right) \subseteq \mathcal{W}$ (resp. $\underline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right) \not\subseteq \mathcal{W}$) lies entirely inside (resp. outside) $\mathcal{Z}_{[p],s}$. We shall refer to the first type of cells as safe and to the second as unsafe. Cells not belonging to either of these two classes intersect with the boundary of $\mathcal{Z}_{[p],s}$ and will be labeled as mixed. Our goal is to approximate $\mathcal{Z}_{[p],s}$ by the union of a finite number of safe cells but, in general, the shape of the robot's configuration space does not admit such representations.

As such, we instead compute a cover of $\mathcal{Z}_{[p],s}$ consisting of both safe and mixed cells, where the size of the mixed cells should be kept as small as possible. To do so, we propose the adaptive subdivision method outlined in Algorithm 2. Beginning with the state space cell defined by the Cartesian product of the axis-aligned bounding box of $\mathcal{W}$ and the set $\mathcal{Z}_{[q]}$, the proposed algorithm builds a partition $\mathcal{P}$ of the robot's state space by adaptively subdividing cells that lie on its boundary $\mathcal{F}$. Specifically, at each iteration, a cell $(\mathcal{C}_{[p]}, \mathcal{C}_{[q]}) \in \mathcal{F}$ is selected and if $\overline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right)$ lies inside $\mathcal{W}$, then it is added to $\mathcal{P}$. Instead, if $\overline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right)$ overlaps with the complement of $\mathcal{W}$, then that cell gets discarded. If, at this point, the cell cannot be labeled either safe or unsafe and its size is not smaller than a user specified threshold ϵ_w, then the cell gets subdivided and the new cells are added to the boundary $\mathcal{F}$. Otherwise, if the cell's size is smaller than the specified threshold and cannot be subdivided any more, it is included in the partition $\mathcal{P}$. Finally, the desired over-approximation $\overline{\mathcal{Z}}_s$ can be obtained as the union of the cells in the partition $\mathcal{P}$.

In the following theorem we show that Algorithm 2 terminates in finite time and its worst case run-time is related to the size of the workspace and the reciprocal of the volume of the smallest allowable cell. Moreover, we show that the resulting cell partition $\mathcal{P}$ lies in the workspace and covers the robot's entire safe set, exceeding it in proportion to a user-selected tolerance. Finally, we bound the possible safety violations that can result from the operation of the robot in $\mathcal{P}$. Specifically, we show that violations are proportional to the size of translation outside of $\mathcal{Z}_s$, but rotations outside of $\mathcal{Z}_s$ are amplified by the maximum distance from any point in the robot to its center of rotation. A proof of Theorem 1 can be found in Vlantis et al. (2022).

Theorem 1 *Suppose a compact set* $\mathcal{W} \subset \mathbb{R}^2$*, a mapping* $\mathcal{R} : \mathbb{R}^2 \times [0, 2\pi) \mapsto \mathfrak{S}(\mathbb{R}^2)$*, and* $\epsilon_w > 0$*. Let also* $\overline{\mathcal{R}}, \underline{\mathcal{R}} : \mathfrak{S}(\mathbb{R}^2 \times [0, 2\pi)) \mapsto \mathfrak{S}(\mathbb{R}^2)$ *be mappings satisfying Equation 7. Let* $\mathcal{F}_0$ *be the initialization of* $\mathcal{F}$*. Then Algorithm 2 terminates in finite time after at most* $2\epsilon_w^{-3}\mathrm{vol}\left(\mathcal{F}_0\right)$ *repetitions of the while loop. It's output,* $\mathcal{P}$*, is a collection of cells in* $\mathcal{W}$ *satisfying* $\mathcal{Z}_s \subseteq \cup_{\mathcal{C}\in\mathcal{P}}\mathcal{C} \subseteq \mathcal{Z}_s \oplus (\mathcal{C}_{\epsilon_w} \times 0)$*. Suppose further that* $\mathcal{R}([x, y, \theta]^\mathsf{T}) = R(\theta)\mathrm{R} + [x, y]^\mathsf{T}$*, where* $R(\theta)$ *is a rotation in* $\mathbb{R}^2$*, and* $\mathrm{R} \subseteq \mathcal{B}_r$ *for some* $r > 0$*. In this case,* $\cup_{\mathcal{C}\in\mathcal{P}}\mathcal{C} \cup_{p\in\mathcal{C}_{[p]}} \mathcal{R}(p) \subseteq \mathcal{W} \oplus \mathcal{B}_v$ *where* $v = 2r^2(1 - \cos(\epsilon_w)) + 2\sqrt{2}\epsilon_w$*.*

3.2. Over-Approximation of the Forward Reachable Set of the Closed-Loop System

Using the over-approximation $\overline{\mathcal{Z}}_s$ of the robot's safe state space as a set of initial robot states, this section employs reachability analysis to over-approximate the closed-loop system's forward reachable set. Since the accuracy of the over-approximation of the forward reachable set depends on the partition $\mathcal{P}$ of the over-approximation of the safe set $\overline{\mathcal{Z}}_s$, we propose to refine the partition $\mathcal{P}$ to improve the accuracy of the over-approximation $\overline{\mathcal{Z}}_c$. This controls the tradeoff between computational complexity affected by the number of cells in $\mathcal{P}$ and accuracy of the over-approximation $\overline{\mathcal{Z}}_c$.

More specifically, to obtain a tight over-approximation of the set of states $\mathcal{Z}_c$ reachable from $\overline{\mathcal{Z}}_s$ using the robot's closed-loop dynamics, we begin by noting that the partition constructed by Algorithm 2 consists of cells defined by n-dimensional intervals. As such, given a state space cell $\mathcal{C} \in \overline{\mathcal{Z}}_s$, the state space cell $\mathfrak{R}\left(\mathcal{C}\right)$ reachable from $\mathcal{C}$ can be computed using F defined in (2) which returns a bounding box enclosing the set of states reachable from $\mathcal{C}$ under the neural network controller, i.e., $\mathfrak{R}\left(\mathcal{C}; w, b\right) = F\left(\mathcal{C}, \Phi\left(\mathcal{C}; w, b\right)\right)$ where Φ is a continuous map of a n-dimensional interval to a m-dimensional interval such that $\phi\left(z; w, b\right) \in \Phi\left(\mathcal{C}; w, b\right), \ \forall z \in \mathcal{C}$. To obtain a function Φ that returns a valid over-approximation of the set of control inputs generated by the

Algorithm 2: Given a robot described by, $\mathcal{R}$, an allowable workspace, $\mathcal{W}$, and a tolerance, ϵ_w, produce a cell collection, $\mathcal{P}$, covering $\overline{\mathcal{Z}}_s$ with bounded safety violations.

```
function $\mathcal{P}$ =BUILDPRTN($\mathcal{W}, \mathcal{R}, \epsilon_w$)
    $\mathcal{P}$, $\mathcal{F}$ $\leftarrow$ $\{\}$, {A bounded cell covering of $\mathcal{W}$} $\times \mathbb{S}^1$ ;
    while $\mathcal{F}$ is not empty
        $\mathcal{C}_{[p]}$ $\leftarrow$ a cell in $\mathcal{F}$ ;
        $\mathcal{F}$ $\leftarrow$ $\mathcal{F} \setminus \mathcal{C}_{[p]}$;
        if $\overline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right) \subseteq \mathcal{W}$
            $\mathcal{P} \leftarrow \mathcal{P} \cup \left\{\mathcal{C}_{[p]} \times \mathcal{Z}_{[q]}\right\}$
        elseif $\underline{\mathcal{R}}\left(\mathcal{C}_{[p]}\right) \subseteq \mathcal{W}$
            if MAXWIDTH($\mathcal{C}_{[p]}$)$< \epsilon_w$
                $\mathcal{F} \leftarrow \mathcal{F}\cup$SUBDIVIDECELL($\mathcal{C}_{[p]}$)
            else
                $\mathcal{P} \leftarrow \mathcal{P} \cup \left\{\mathcal{C}_{[p]} \times \mathcal{Z}_{[q]}\right\}$
            end
        end
    end
end
```

neural network ϕ, we employ the Interval Bound Propagation (IBP) method Xiang et al. (2018). In other words, for each cell $\mathcal{C} \in \mathcal{P}$, we employ the IBP method (function Φ) to compute bounds on the robot's control inputs, which we then propagate (function F) to obtain bounds on the robot's forward reachable set. Thus, the over-approximation $\overline{\mathcal{Z}}_c$ of the set of states reachable by the closed-loop system after one step can be obtained as $\overline{\mathcal{Z}}_c = \bigcup\limits_{\mathcal{C}\in\mathcal{P}} \mathfrak{R}\left(\mathcal{C}; w, b\right)$.

We remark that the over-approximation error between the bounds on the control inputs computed using IBP and the actual bounds on the control inputs generated by ϕ for the states in $\mathcal{C}$ increases with the size of $\mathcal{C}$, as explained in Xiang et al. (2018). Therefore, a fine partition $\mathcal{P}$ of the safe space over-approximation $\overline{\mathcal{Z}}_s$ is generally required in order to obtain a tight over-approximation of $\mathcal{Z}_c$. To refine $\mathcal{P}$ while keeping the total number of cells as low as possible, we further subdivide only cells whose forward reachable set intersects with the complement of $\overline{\mathcal{Z}}_s$. To identify such cells, we recall that the forward reachable set $\mathfrak{R}\left(\mathcal{C}\right)$ of $\mathcal{C}$ is a composite interval with the same dimension as $\mathcal{C}$ and consider the two components $\mathcal{C}'_{[p]} \in \mathcal{Z}_{[p]}$ and $\mathcal{C}'_{[q]} \in \mathcal{Z}_{[q]}$ of $\mathfrak{R}\left(\mathcal{C}\right)$ such that $\mathfrak{R}\left(\mathcal{C}\right) = \mathcal{C}'_{[p]} \times \mathcal{C}'_{[q]}$. Following the procedure presented in subsection 3.1, we can compute an over-approximation $\overline{\mathcal{R}}\left(\mathcal{C}'_{[p]}\right)$ of the robot's footprint corresponding to the forward reachable set of configurations $\mathcal{C}'_{[p]}$. Therefore, a cell $\mathcal{C}$ in $\mathcal{P}$ only needs to be subdivided if $\overline{\mathcal{R}}\left(\mathcal{C}'_{[p]}\right)$ intersects with the complement of the workspace $\mathcal{W}$.

The above process is outlined in Algorithm 3 which adaptively subdivides cells in $\mathcal{P}$ with large over-approximation errors of their forward reachable sets. Specifically, for each cell $\mathcal{C} = (\mathcal{C}_{[p]}, \mathcal{C}_{[q]})$ in $\mathcal{P}$, we check whether the area of $\mathrm{cmpl}(\mathcal{W})$ covered by $\overline{\mathcal{R}}\left(\mathcal{C}'_{[p]}\right)$ or the volume of $\mathrm{cmpl}\left(\mathcal{Z}_{[q]}\right)$ covered by $\mathcal{C}'_{[q]}$ are greater than user specified thresholds ϵ_{V_p} and ϵ_{V_q}, respectively. If these conditions hold and the size of the cell admits further subdivision, then the cell gets split and replaced by smaller ones. Otherwise, the sibling $\mathcal{C}^{sib}$ of cell $\mathcal{C}$ is retrieved[1] and if the volume of their parent cell $\mathcal{C}_{mrg}$ that lies outside the set of safe state $\mathcal{Z}_s$ is negligible, then the cells $\mathcal{C}$ and $\mathcal{C}^{sib}$ are

1. Notice that, since each cell can be split only once into two subcells, $\mathcal{P}$ can be represented as a binary tree.

Algorithm 3: Refine $\mathcal{P}$ and remove cells where the robot's one-step reachable set violates safety beyond specified tolerances.

```
function $\mathcal{P}'$ =REFINEPRTN($\mathcal{P}$, $\epsilon_w$, $\epsilon_{V_p}$, $\epsilon_{V_q}$; $\mathcal{W}$, $\mathcal{R}$, $\mathcal{Z}_{[q]}$, $f$)
    $\mathcal{F}$, $\mathcal{P}' \leftarrow \mathcal{P}$, $\emptyset$ ;
    while $\mathcal{F}$ is not empty
        $\mathcal{C} \leftarrow$ a leaf cell in the last level of $\mathcal{F}$ that has not yet been examined;
        if PENALTYTEST($\mathcal{C}$, $\epsilon_{V_p}$, $\epsilon_{V_q}$; $\mathcal{W}$, $\mathcal{R}$, $\mathcal{Z}_s$, $f$)
            if MAXWIDTH($\mathcal{C}_{[p]}$)$> \epsilon_w$
                $\mathcal{F} \leftarrow \mathcal{F} \cup \left(\text{SUBDIVIDECELL}(\mathcal{C}_{[p]}) \times \mathcal{C}_{[q]}\right)$ ;        // Children of $\mathcal{C}$
            else
                $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{C}$ ;
            end
        else
            $\mathcal{C}^{sib}$, $\mathcal{C}^{par} \leftarrow$ GETSIBLING($\mathcal{C}$), GETPARENT($\mathcal{C}$);
            if PENALTYTEST($\mathcal{C}^{sib}$, $\epsilon_{V_p}$, $\epsilon_{V_q}$; $\mathcal{W}$, $\mathcal{R}$, $\mathcal{Z}_s$, $f$)
                if MAXWIDTH($\mathcal{C}^{sib}_{[p]}$)$< \epsilon_w$
                    $\mathcal{F} \leftarrow \mathcal{F} \cup \left(\text{SUBDIVIDECELL}(\mathcal{C}^{sib}_{[p]}) \times \mathcal{C}^{sib}_{[q]}\right)$ ;        // Children of $\mathcal{C}^{sib}$
                else
                    $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{C}$ ;
                end
            elseif $\mathcal{C}^{sib} = \emptyset$ or CANNOTMERGE($\mathcal{C}, \mathcal{C}^{sib}$)
                $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{\mathcal{C}, \mathcal{C}^{sib}, \mathcal{C}_{par}\})$,   $\mathcal{P}' \leftarrow \mathcal{P}' \cup \mathcal{C} \cup \mathcal{C}^{sib}$ ;
            else
                $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{\mathcal{C}, \mathcal{C}^{sib}, \mathcal{C}_{par}\}) \cup$ MERGE($\mathcal{C}^{sib}$, $\mathcal{C}$) ;        // Replace $\mathcal{C}_{par}$
            end
        end
    end
end
```

merged to reduce the size of $\mathcal{P}$. Finally, the over-approximation $\overline{\mathcal{Z}}_c$ is obtained as the union of the forward reachable set of each cell in $\mathcal{P}$. Algorithm 3 effectively controls the trade-off between computational complexity affected by the number of cells in the partition $\mathcal{P}$ and accuracy of the over-approximation $\overline{\mathcal{Z}}_c$.

In the following theorem we show that Algorithm 3 terminates in finite time and its worst case run-time is related to the size of the workspace, the number of intervals in $\mathcal{Z}_{[q]}$, and the reciprocal of the volume of the smallest allowable cell. Moreover, we show that the resulting cell partition $\mathcal{P}'$ inherits the safety assurances of $\mathcal{P}$ and that safety violations of its one-step reachable set are bounded by a user-specified tolerance. A proof of Theorem 2 can be found in Vlantis et al. (2022).

Theorem 2 *Suppose Algorithm 2 is applied to a compact set* $\mathcal{W} \subset \mathbb{R}^2$ *and let* $\mathcal{P} = \mathcal{P}_{[p]} \times \mathcal{Z}_{[q]}$ *denote its output. Consider also the mapping* $\mathcal{R} : \mathbb{R}^2 \times [0, 2\pi) \mapsto \mathfrak{S}(\mathbb{R}^2)$, *the tolerance* $\epsilon_w > 0$, *and assume that the assumptions of Theorem 1 hold. Suppose that Algorithm 3 is applied to* $\mathcal{P}$ *with discrete-time forward dynamics governed by Equations 1-4 and tolerances* $\epsilon_{V_p}, \epsilon_{V_q} > 0$. *Then, Algorithm 3 terminates after* $\#\left(\mathcal{Z}_{[q]}\right) \operatorname{vol}\left(\mathcal{P}_{[p]}\right) \epsilon_w^{-3}$ *repetitions of its while loop and the output,* $\mathcal{P}'$, *satisfies* $\cup_{\mathcal{C} \in \mathcal{P}'} \mathcal{C} \subseteq \mathcal{Z}_s \oplus (\mathcal{C}_{\epsilon_w} \times 0)$ *and* $\cup_{\mathcal{C} \in \mathcal{P}'} \mathcal{C} \cup_{p \in \mathcal{C}_{[p]}} \mathcal{R}(p) \subseteq \mathcal{W} \oplus \mathrm{B}_v$. *Moreover, if* $\mathcal{C}' = \mathfrak{R}(\mathcal{C})$ *for* $\mathcal{C} \in \mathcal{P}'$, *then* $\operatorname{vol}\left(\overline{\mathcal{R}}\left(\mathcal{C}'_{[p]}\right) \cap \operatorname{cmpl}(\mathcal{W})\right) \leq \epsilon_{V_p}$ *and* $\operatorname{vol}\left(\mathcal{C}'_{[q]} \cap \operatorname{cmpl}\left(\mathcal{Z}_{[q]}\right)\right) \leq \epsilon_{V_q}$.

Algorithm 4: Check whether the outer approximation of the robot's one-step reachable set violates safety by more than margins ϵ_{V_p}, ϵ_{V_q} anywhere in cell $\mathcal{C}$.

```
function $A$ =PenaltyTest($\mathcal{C}$, $\epsilon_{V_p}$, $\epsilon_{V_q}$; $\mathcal{W}$, $\mathcal{R}$, $\mathcal{Z}_{[q]}$, $f$)
    $A$, $\mathcal{C}' \leftarrow$ false, $\mathfrak{R}(\mathcal{C})$ ;
    if vol $\left(\overline{\mathcal{R}}\left(\mathcal{C}'_{[p]}\right) \cap \mathrm{cmpl}(\mathcal{W})\right) > \epsilon_{V_p}$ or vol $\left(\mathcal{C}'_{[q]} \cap \mathrm{cmpl}\left(\mathcal{Z}_{[q]}\right)\right) > \epsilon_{V_q}$
        $A \leftarrow$ true ;
    end
end
```

3.3. Safety-Aware Control Training

This section uses the over-approximation of the forward reachable set of the closed loop system $\overline{\mathcal{Z}}_c$, to design appropriate penalty terms which, when added to the loss function (6), minimize the overlap between this forward reachable set and the set of unsafe states. In this way, we can reduce safety violations after re-training of the neural network controller (3). To do so, we solve the following optimization problem at every iteration of Algorithm 1 to update the neural network parameters

$$\underset{w,b}{\text{minimize}} \quad J^{\star}(w,b) \tag{8}$$

$$J^{\star}(w,b) = J(w,b) + \lambda_S \cdot h(w,b), \tag{9}$$

where λ_S is a positive constant and h a penalty term that measures safety violations. The goal in designing the penalty term h is to push the over-approximation of the new forward reachable set $\overline{\mathcal{Z}}_c$ inside the under-approximation $\underline{\mathcal{Z}}_s = \cup_{\mathcal{C}\in\mathcal{P}_S}\mathcal{C}$ of the set of safe states $\mathcal{Z}_s$, where $\mathcal{P}_S$ denotes the subset of the partition $\mathcal{P}$ consisting of only safe cells. To do so, we define the penalty term h as $h(w,b) = \sum_{\mathcal{C}'\in\mathfrak{R}(\mathcal{P})} V\left(\mathcal{C}', \mathcal{P}_S\right)^2$, where $\mathfrak{R}(\mathcal{P}) = \{\mathfrak{R}(\mathcal{C}) \mid \mathcal{C}\in\mathcal{P}\}$ and $V(\mathcal{C}, \mathcal{P}_S)$ is a valid metric of the volume occupied by the intersection of the composite interval $\mathcal{C}$ and $\cup_{\mathcal{C}\in\mathcal{P}_S}\mathcal{C}$. Thus, h vanishes only if $\mathcal{C}' \subseteq \cup_{\mathcal{C}\in\mathcal{P}_S}\mathcal{C}$ for all $\mathcal{C}' \in \mathfrak{R}(\mathcal{P})$, i.e., the set of safe states is rendered invariant under the closed-loop dynamics. Note that the overlap between the forward reachable set and the set of unsafe states $\mathcal{Z}_c \setminus \mathcal{Z}_s$ also provides numerical bounds on possible safety violations by the closed loop system, that can be used as a measure of reliability of the neural network controller (3).

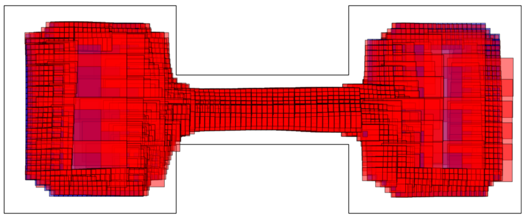
4. Numerical Experiments

This section illustrates the proposed method through numerical experiments involving a rectangular robot operating inside an H-shaped workspace shown in Figure 2. The robot is assumed to obey the simple, holonomic kinematic model $z_{k+1} = z_k + K\cdot\phi(z_k)$, with $z = p$ and $K = 0.01$. The dataset $\mathcal{D}$ was assembled by computing 500 feasible trajectories using the RRT* algorithm Karaman and Frazzoli (2010) and initializing the robot at random configurations. Particularly, each point $(z,u) \in \mathcal{D}$ corresponds to a sampled trajectory point (z,u'), where the control input u' is modified to ensure that the implicitly defined vector field vanishes at the goal configuration, i.e., $u = 10\cdot\frac{u'}{\|u'\|}\frac{\|z^{\star}-z\|}{1-\|z^{\star}-z\|}$. The over-approximation of the robot's forward reachable set for each cell $\mathcal{C}$ is computed as the Minkowski sum between that cell and the controller's output reachable set for that cell, i.e., $F(\mathcal{C}, \Phi(\mathcal{C})) = \mathcal{C} \oplus (K\cdot\Phi(\mathcal{C}))$. To evaluate the penalty term, h, the metric is defined by $V(\mathcal{C}, \mathcal{P}_S) = (\mathrm{vol}_\delta(\mathcal{C}))^{1/3} - \left(\sum_{\mathcal{C}'\in\mathcal{P}_S}\mathrm{vol}_\delta(\mathcal{C}\cap\mathcal{C}')\right)^{1/3}$, where $\delta = [\delta_x, \delta_y, \delta_\theta] = [1, 1, 1/2\pi]$

are constant scaling factors. Particularly, the cubic root of the volume of composite intervals in $V(\mathcal{C}, \mathcal{P}_S)$ is used to prevent terms corresponding to slim cells from vanishing too fast when, e.g., only one of their dimensions has grown small whereas the others have not. The scaling factors $\delta_x, \delta_y, \delta_\theta$ admit different weights to each dimension, given that they have incompatible units.

Three separate neural network controllers ϕ_1, ϕ_2, ϕ_3 with various shapes were trained to safely steer the robot to the desired configuration using the proposed method with different subdivision thresholds ϵ_w; for details on the selection of parameters see Vlantis et al. (2022). After initially training each network using the data in $\mathcal{D}$, we used Algorithm 2 to obtain an initial partition $\mathcal{P}$ of the robot's configuration space. Then, we applied Algorithm 3 to refine the initial partition $\mathcal{P}$ based on the system's forward reachable sets. Table 1 of Vlantis et al. (2022) shows the number of cells in the partition $\mathcal{P}$ associated with each neural network controller before and after the first refinement. We remark that blindly partitioning the configuration space $\mathcal{W} \times \mathbb{S}^1$ into cells with dimensions $0.1 \times 0.1 \times 0.2\pi$ would result in 14000 cells or more, but that the proposed method requires $1 - 6990/14000 \approx 50\%$ and $1 - 5536/14000 \approx 60\%$ fewer subdivisions for ϕ_1 and ϕ_3, respectively. Finally, 50 refinement and update iterations of of Algorithm 1 were performed.

Example projections of the cells in each partition onto the xy-plane along with the over - approximations of their forward reachable sets before and after retraining can be seen in Figure 2. Further projections and sample trajectories can be found in Vlantis et al. (2022). We remark that the area covered by the over-approximation of the robot's forward reachable set (colored in red) that lies outside the set of initial states (colored in blue) has noticeably decreased at the end of 50-th epoch, especially for the case of the neural network controller ϕ_1. This area provides spatial bounds on possible safety violations induced by the trained controllers.


(a) ϕ_1: Before re-training

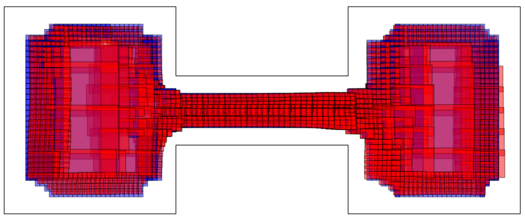

(b) ϕ_1: After re-training

Figure 2: Projections of cells (blue boxes) of partition corresponding to neural network controller ϕ_1 onto the xy-plane overlayed with the over-approximations of their forward reachable sets (red boxes) before and after retraining. Cell projections for ϕ_2 and ϕ_3 are visualized in Vlantis et al. (2022).

5. Conclusions

This work addresses the problem of safely steering a polygonal robot operating inside a compact workspace to a desired configuration using a feed-forward neural network controller trained to avoid collisions between the robot and the workspace boundary. By dividing the safe region into cells, interval analysis admits computationally tractable inner and outer approximations of the reachable set despite nonlinear dynamics. Compared to existing methods that train neural network controllers with closed-loop safety guarantees, our approach lifts burdensome data density requirements near the boundary of the safe state space, and instead allows for *graceful* safety violations, i.e., of a bounded magnitude that can be spatially controlled. Future work will incorporate further experiments incorporating more complex environments and comparing to existing methods, including Lin et al. (2020); Yang et al. (2022), techniques to limit constraint violations over multi-step trajectories, and generalized cell refinement and re-combination schemes to include robots moving in three dimensional environments.

Acknowledgments

This work is supported in part by AFOSR under award #FA9550-19-1-0169, by ONR under agreement #N00014-18-1-2374, and by NSF under award CNS-1932011.

References

Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J. Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments, 2019. URL https://arxiv.org/abs/1905.00532.

Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3387–3395, 2019. doi: 10.1609/aaai.v33i01.33013387. URL https://doi.org/10.1609/aaai.v33i01.33013387.

Souradeep Dutta and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In *ACM International Conference on Hybrid Systems: Computation and Control*, pages 157–168, 04 2019. doi: 10.1145/3302504.3311807.

Souradeep Dutta, Susmit Jha, Sriram Sanakaranarayanan, and Ashish Tiwari. Output range analysis for deep neural networks, 2017. URL https://arxiv.org/abs/1709.09130.

Jaime F. Fisac, Anayo K. Akametalu, Melanie N. Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems, 2018. URL https://arxiv.org/abs/1705.01292.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Int. Res.*, 24(1):81–108, July 2005. ISSN 1076-9757.

Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming, 2020. URL https://arxiv.org/abs/2004.07876.

Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems, 2019. URL https://arxiv.org/abs/1906.10654.

Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning, 2010. URL https://arxiv.org/abs/1005.0416.

Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks, 2017. URL https://arxiv.org/abs/1702.01135.

Anjian Li, Somil Bansal, Georgios Giovanis, Varun Tolani, Claire Tomlin, and Mo Chen. Generating robust supervision for learning-based visual navigation using hamilton-jacobi reachability, 2020. URL https://arxiv.org/abs/1912.10120.

Xiao Li and Calin Belta. Temporal logic guided safe reinforcement learning using control barrier functions, 2019. URL https://arxiv.org/abs/1903.09885.

Xuankang Lin, He Zhu, Roopsha Samanta, and Suresh Jagannathan. Art: abstraction refinement-guided training for provably correct neural networks. In *Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2020.

Lars Lindemann, Haimin Hu, Alexander Robey, Hanwen Zhang, Dimos V. Dimarogonas, Stephen Tu, and Nikolai Matni. Learning hybrid control barrier functions from data, 2020. URL https://arxiv.org/abs/2011.04112.

Motoya Ohnishi, Li Wang, Gennaro Notomista, and Magnus Egerstedt. Barrier-certified adaptive reinforcement learning with applications to brushbot navigation. *IEEE Transactions on Robotics*, 35(5):1186–1205, 2019. doi: 10.1109/tro.2019.2920206. URL https://doi.org/10.1109/tro.2019.2920206.

Theodore Perkins and Andrew Barto. Lyapunov design for safe reinforcement learning control. *J. Mach. Learn. Res.*, 3:803–, 05 2003. doi: 10.1162/jmlr.2003.3.4-5.803.

Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V. Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations, 2020. URL https://arxiv.org/abs/2004.03315.

Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees, 2018. URL https://arxiv.org/abs/1805.02242.

Xiaowu Sun and Yasser Shoukry. Provably correct training of neural network controllers using reachability analysis, 2021. URL https://arxiv.org/abs/2102.10806.

Panagiotis Vlantis, Leila J. Bridgeman, and Michael M. Zavlanos. Failing with grace: Learning neural network controllers that are boundedly unsafe, 2022. URL https://arxiv.org/abs/2106.11881.

Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson. Output reachable set estimation and verification for multi-layer neural networks, 2018. URL https://arxiv.org/abs/1708.03322.

Xiaodong Yang, Tom Yamaguchi, Hoang-Dung Tran, Bardh Hoxha, Taylor T. Johnson, and Danil Prokhorov. Neural network repair with reachability analysis. In Sergiy Bogomolov and David Parker, editors, *Formal Modeling and Analysis of Timed Systems*, pages 221–236, Cham, 2022. Springer International Publishing. ISBN 978-3-031-15839-1.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions, 2018. URL https://arxiv.org/abs/1811.00866.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks, 2019. URL https://arxiv.org/abs/1906.06316.

D. J. Zhu and J. . Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7(1):9–20, Feb 1991. ISSN 1042-296X. doi: 10.1109/70.68066.