FI-ODE: Certified and Robust Forward Invariance in Neural ODEs

Yujia Huang [*1] Ivan Dario Jimenez Rodriguez [*1] Huan Zhang [2] Yuanyuan Shi [3] Yisong Yue [1]

Abstract

Forward invariance is a long-studied property in control theory that is used to certify that a dynamical system stays within some pre-specified set of states for all time, and also admits robustness guarantees (e.g., the certificate holds under perturbations). We propose a general framework for training and provably certifying robust forward invariance in Neural ODEs. We apply this framework in two settings: certified adversarial robustness for image classification, and certified safety in continuous control. Notably, our method empirically produces superior adversarial robustness guarantees compared to prior work on certifiably robust Neural ODEs (including implicit-depth models).

1. Introduction

Deployment of neural networks in the real-world increasingly demands formal certificates of performance. Examples of certified behaviors include correctly classifying samples despite perturbations on the neural network inputs (Wong & Kolter, 2018b; Raghunathan et al., 2018; Cohen et al., 2019), and controlling a dynamical system to stay inside a safe set (Jin et al., 2020). We often need provable certificates given that even impressive empirical robustness often fails under unforeseen stronger attacks (Athalye et al., 2018), and we desire assurances that can be useful in applications with strong reliability requirements.

Forward invariance has been extensively used in control theory to certify dynamical systems for safety (Ames et al., 2016) and robustness under adversarial perturbations (Khalil et al., 1996). To repurpose this concept for Machine Learning, we focus on the Neural Ordinary Differential Equation (NODE) function class (Haber & Ruthotto, 2017; E, 2017; Chen et al., 2018), which is a natural starting point for incorporating control-theoretic tools (cf. (Yan et al., 2020; Kang et al., 2021; Liu et al., 2020; Jimenez Rodriguez et al., 2022)). In this new setting, forward invariance guarantees that NODE trajectories never leave a desirable set which can produce robustness guarantees for classification or safety guarantees for control.

Our contributions: We present FI-ODE, a general approach for training certifiably forward invariant NODEs. We provide a robustness guarantee using the forward invariance of sub-level sets of Lyapunov functions. One can train a NODE such that a task-specific cost function (e.g., cross-entropy loss in image classification, or state-based cost in continuous control) becomes the Lyapunov function for the ODE. We train with a variant of Lyapunov training (Jimenez Rodriguez et al., 2022) that focuses on states that are crucial for certifying robust forward invariance. To make certification practical, we constrain the hidden states of a NODE to evolve on a compact set by projecting the dynamics of NODE to satisfy certain barrier conditions. We provably verify our method through a combination of efficient sampling and a new interval propagation technique compatible with optimization layers.

We empirically show superior ℓ_2 certified robustness compared to all other certifiably robust ODE-based models (including implicit depth models such as Deep-Equilibrium models (Chen et al., 2021)) on MNIST and CIFAR-10. We also demonstrate the generality of our approach by applying it to nonlinear control problems, where we train and certify NODE policies to keep the system within a safe region. Our code is avaiable at `https://github.com/yjhuangcd/FI-ODE.git`.

2. Preliminaries

Consider training data of the form $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} \in \mathbb{R}^m$ is the input, and $\boldsymbol{y} \in \mathbb{R}^n$ is a prediction target (e.g., a classification label, or a desired state for a controller). Let $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^l$ denote the parameters of the learned model.

Neural ODE (NODE) Model Class.

$$\boldsymbol{\eta}(0) = \boldsymbol{\eta}_0, \qquad \text{(initial condition)} \tag{1a}$$

$$\frac{d\boldsymbol{\eta}}{dt} = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}(t), \boldsymbol{x}) \quad \text{(continuum of hidden layers).} \tag{1b}$$

The hidden states of the NODE are $\boldsymbol{\eta} \in \mathcal{H} \subset \mathbb{R}^n$ where $\mathcal{H}$ is compact and connected. In general, we assume the overparameterized setting, where $\boldsymbol{\theta}$ is expressive enough to fit the dataset.

[*]Equal contribution [1]Caltech [2]Carnegie Mellon University [3]UCSD. Correspondence to: Yujia Huang <yjhuang@caltech.edu>, Ivan Dario Jimenez Rodriguez <ivan.jimenez@caltech.edu>.

In order to make predictions, one needs to define a mapping from $\boldsymbol{\eta}$ to the prediction output $\boldsymbol{y}$. We consider the special case where that mapping is the identity function.[1] The system evolves over $t \in [0, T]$ and, although $\boldsymbol{\eta}(t)$ can be evaluated on any t in this range, the final prediction is $\boldsymbol{\eta}(T)$. Thus, the goal is to train a NODE such that $\boldsymbol{\eta}(T) = \boldsymbol{y}$.

2.1. Forward Invariance

Forward Invariance refers to sets of states of a dynamical system (e.g., Equation (1b)) where the system can enter but never leave. Formally:

Definition 1 (Forward Invariance). *A set $\mathcal{S} \subseteq \mathcal{H}$ is forward invariant if $\boldsymbol{\eta}(t) \in \mathcal{S} \Rightarrow \boldsymbol{\eta}(t') \in \mathcal{S}, \forall t' \geq t$.*

Forward invariance can be applied generally in NODEs: we can choose the dynamics in Equation (1b) to render almost any set we choose forward invariant. In classification we will be concerned with the set of states that produce a correct classification, while in control we want to keep the system safe. In both cases, we start from a set and then shape the dynamics to achieve forward invariance of that set.

Trajectory-wise versus Point-wise Analysis. Figure 1 depicts two ways of analyzing forward invariance. On the left we consider entire trajectories that result from running the ODE (i.e., running the forward pass) and determine forward invariance by checking whether the trajectories leave the target set (whose boundary is depicted by the yellow line). Such trajectory-level analyses are computationally expensive due to running ODE integration to generate trajectories. This approach also poses a challenge for verification since trajectories can only be integrated for finite time T and the dynamics may be close to leaving the set shortly thereafter ($T + \varepsilon$), which translates into vulnerability to perturbations.

An alternative approach, depicted on the right in Figure 1, relies on point-wise conditions: we look at the dynamics point-wise over the state-space and infer whether the set above the yellow line is forward invariant. In this situation robust certification can be significantly easier because we only need to verify that perturbations to the dynamics are point-wise still pointing in the right direction, rather than analyzing the effect of perturbed dynamics over an entire trajectory (i.e., we do not need to do ODE integration).

Lyapunov Sublevel Sets. W use Lyapunov potential functions from control theory to define sets to render forward invariant. A potential function $V : \mathcal{H} \rightarrow \mathbb{R}_{\geq 0}$ is a Lyapunov function for the ODE if for all reachable states η we have:

$$\frac{d}{dt} V(\boldsymbol{\eta}(t)) \leq -\kappa(V(\boldsymbol{\eta})). \tag{2}$$

[1]This is analogous to the final layer of a standard neural network being the identity matrix, which typically does not lead to a meaningful reduction in model expressiveness.

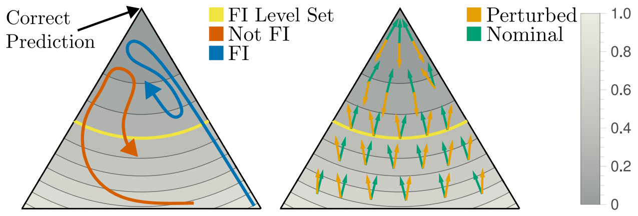

Figure 1. Showing depictions of trajectories (Left) and dynamics (Right) of the state-space of a NODE. The top corner corresponds to the correct prediction. The contours show a quadratic potential function centered on the correct prediction. The yellow-line is the upper bound of a forward invariant sublevel set of the potential function. Left: visualizing trajectories (i.e., full inferences of the NODE that either violate (orange) or satisfy (blue) the forward invariance condition. Right: visualizing the flow field (i.e., dynamics) of the NODE, including under both nominal and perturbed inputs. In this example, the perturbed flow field is still generates trajectories that satisfy forward invariance, meaning that the forward invariance certificate is robust to perturbations.

Here $\kappa : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is a so-called class $\mathcal{K}_\infty$ function, which means that it is strictly increasing, $\kappa(0) = 0$, and $\lim_{r \rightarrow \infty} \kappa(r) = \infty$. Intuitively, Equation (2) implies a V that decreases exponentially quickly, which we formalize in Theorem B.1 that was first introduced in Ames et al. (2014).

A Lyapunov sublevel set is the set of states $\boldsymbol{\eta}$ where $V(\boldsymbol{\eta}) \leq c$ for some constant c. Since a Lyapunov function V is always decreasing in time, once a state enters a Lyapunov sublevel set it remains there for all time. The potential function depicted in Figure 1 can be viewed as a Lyapunov function, and the yellow line the boundary of the corresponding Lyapunov sublevel set.

In practice, one often chooses V to be a standard training loss, such as cross entropy loss for classification. Assuming the NODE is trained such that V is a Lyapunov function for the NODE (discussed in Section 2.3), then one can define an appropriate sublevel set to characterize (via forward invariance) always making the correct prediction.

2.2. Connections to Adversarial Robustness

Suppose $\boldsymbol{\epsilon} \in \mathbb{R}^n$ with $\|\boldsymbol{\epsilon}\| \leq \bar{\epsilon}$ adversarially corrupts the input part of the input-output pair $(\boldsymbol{x}, \boldsymbol{y})$. Certified robustness (Wong & Kolter, 2018a) requires that if $\boldsymbol{x}$ produces a correct prediction then $\boldsymbol{x} + \boldsymbol{\epsilon}$ also produces a correct prediction (e.g., the correct label in multi-class classification). The standard approach for training certifiably robust models bounds the output range with $\bar{\epsilon}$ (Pabbaraju et al., 2020; Lopez et al., 2022), which requires integrating the NODE. In contrast, our approach uses a point-wise analysis. We train and verify that both the nominal and the perturbed dynamics of states at the boundary of the desired set point inwards. This implies that both the nominal and perturbed trajectories will stay within the desired set (Figure 1, right). This allows us to only enforce conditions on the point-wise dynamics

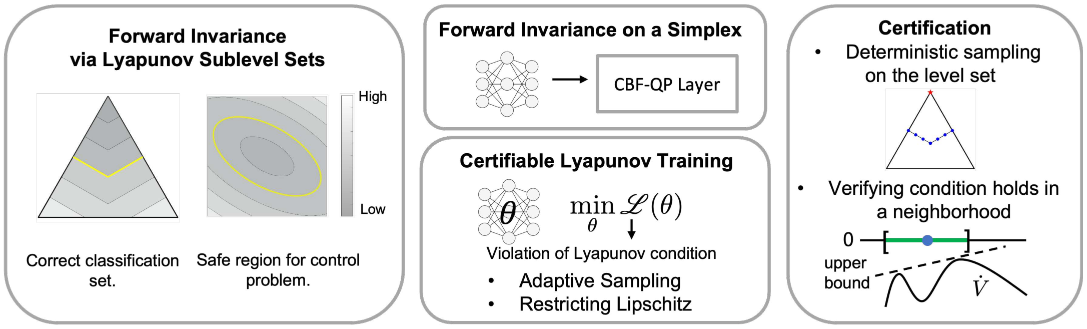
Forward Invariance via Lyapunov Sublevel Sets
High
Low
Correct classification set.
Safe region for control problem.
Forward Invariance on a Simplex
CBF-QP Layer
Certifiable Lyapunov Training
$\min_\theta \mathscr{L}(\theta)$
Violation of Lyapunov condition
Adaptive Sampling
Restricting Lipschitz
Certification
Deterministic sampling on the level set
Verifying condition holds in a neighborhood
0
upper bound
$\dot{V}$

Figure 2. Overview of our FI-ODE framework. We first pick a Lyapunov function based on the shape of the forward invariant set: the boundaries of the Lyapunov sublevel sets are parallel to the boundary of the forward invariant set. For efficient training and certification, we evolve the system on a compact set (e.g. probability simplex). We train the dynamics to satisfy Lyapunov conditions via certifiable Lyapunov training. To certify the forward invariance property, we sample points on the boundary of the forward invariant set and verify conditions hold everywhere on the boundary.

without considering the integration time.

2.3. Lyapunov Training of NODEs

One way to train NODEs to satisfy the Lyapunov condition in Equation (2) is to define a loss that penalizes point-wise violations of it. This idea was first developed in the LyaNet framework (Jimenez Rodriguez et al., 2022), which uses a hinge-like loss on violations of Equation (2).

Definition 2 (Lyapunov Loss (Jimenez Rodriguez et al., 2022)). *For the ODE defined in Equations* (1a) *and* (1b) *and a dataset of input-output pairs* $(\boldsymbol{x}, \boldsymbol{y}) \sim D$ *the Lyapunov Loss is defined as* $\mathscr{L}(\boldsymbol{\theta}) :=$

$$\mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim D}\left[\int_0^T \max\left\{0, \frac{\partial V}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) + \kappa V(\boldsymbol{\eta})\right\} dt\right]. \tag{3}$$

It is straightforward to check that Equation (3) being zero implies satisfying Equation (2). In practice, we employ sampling to approximate the internal integral in Equation (3). It turns out that for certified robust forward invariance, it is important to focus the sampling along the boundary of the Lyapunov sublevel set, which we discuss in Section 3.3.

3. Forward Invariance for NODEs

We now present our FI-ODE framework to enforce forward invariance on NODEs (see Figure 2). We define forward invariance using Lyapunov sublevel sets, and show that robust forward invariance implies a standard adversarial robustness condition for classification (Section 3.1). To enforce forward invariance, we first train to encourage the Lyapunov conditions to hold on the boundary of the target Lyapunov sublevel set (e.g., the yellow line in Figure 1), and then verify. For the verification algorithm to run in a finite time, we constrain the state space to be a compact set (Section 3.2). We train our model using certifiable Lyapunov training (Section 3.3). Building upon the LyaNet framework (Jimenez Rodriguez et al., 2022), we develop an adaptive sampling strategy that focuses on the crucial region of the state space for forward invariance, and control the Lipschitz constant of our dynamics. However, the empirical Lyapunov loss is based on a finite sample, so minimizing it does not guarantee satisfying the Lyapunov condition. Therefore, we develop certification tools to verify the Lyapunov conditions everywhere in the region of interest (Section 3.4).

3.1. Robust FI Implies Adversarial Robustness

Forward invariance for correct classification. Focusing on multi-class classification in this subsection, we first define correct classification for NODEs. For an input $\boldsymbol{x}$ with label y, the output of a NODE after integrating for T time is $\boldsymbol{\eta}(T)$. If $y = \operatorname{argmax} \boldsymbol{\eta}(T)$, we say the NODE *correctly classifies* $\boldsymbol{x}$. We define the correct classification region for class y to be $\mathcal{S}_y = \{\boldsymbol{\eta} | \boldsymbol{\eta} \in \triangle, y = \operatorname{argmax} \boldsymbol{\eta}\}$ (green region in Figure 3(a)), where $\triangle$ stands for the n-class probability simplex: $\{\boldsymbol{\eta} \in \mathbb{R}^n | \sum_{i=1}^n \boldsymbol{\eta}_i = 1, \boldsymbol{\eta}_i \geq 0\}$. Then for a NODE starting with $\boldsymbol{\eta}(0) = \mathbb{1}\frac{1}{n}$, a sufficient condition for correctly classifying $\boldsymbol{x}$ is that the trajectories never escape from $\mathcal{S}_y$, i.e., $\mathcal{S}_y$ should be a forward invariant set.

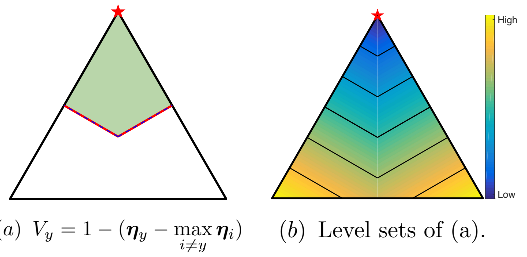

Figure 3. The Lyapunov function and its level sets on a 3-class probability simplex. The red star is the correct class.

Forward invariance criteria. To make $\mathcal{S}_y$ forward invariant, first, we need to make $\boldsymbol{\eta}$ evolve in the probability simplex $\triangle$ (Section 3.2), since enforcing $\boldsymbol{\eta}$ to evolve on a compact set is required for practical certification (and the simplex is a natural choice). Second, we need to make $\{\boldsymbol{\eta} | y = \operatorname{argmax} \boldsymbol{\eta}\}$ forward invariant. We achieve this by defining a (Lyapunov) potential function whose level sets are parallel to the decision boundary:

$$V_y = 1 - (\boldsymbol{\eta}_y - \max_{i \neq y} \boldsymbol{\eta}_i) \tag{4}$$

Usually, Lyapunov stability uses a potential function to prove the stability of a given dynamical system. In our setting, the potential function is pre-defined to be positive definite (Equation (4)), and we find a dynamical system that is stable with respect to this potential function (i.e. making this potential function a Lyapunov function). This is possible because the NODEs are typically overparameterized.

We first check that V_y is positive definite: since $0 \leq \eta_i \leq 1$ for all i, we have $V_y \geq 0$. In addition, $V_y = 0$ only when $\eta_y = 1$ and $\eta_i = 0$ for $i \neq y$. Furthermore, the 1-level set of V_y, $\{\boldsymbol{\eta} \in \triangle | V_y(\boldsymbol{\eta}) = 1\}$ equals the decision boundary $\mathcal{D}_y = \{\boldsymbol{\eta} \in \triangle | \boldsymbol{\eta}_y = \max_{i \neq y} \boldsymbol{\eta}_i\}$. Then according to Nagumo's theorem (Nagumo, 1942), if the following condition holds for all $\boldsymbol{\eta} \in \mathcal{D}_y$, then $\mathcal{S}_y$ is forward invariant:

$$\dot{V}_y = \frac{\partial V_y}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) \leq 0. \tag{5}$$

Using Lyapunov training (Jimenez Rodriguez et al., 2022), we can train the NODE dynamics to satisfy this condition.

Robust forward invariance for robust classification. Robust classification says that if a model can correctly classify $\boldsymbol{x}$, it can also correctly classify $\boldsymbol{x} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is some bounded perturbation. We showed earlier that the dynamics $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x})$ must satisfy Equation (5) for correct classification. To ensure robust forward invariance, the dynamics for the perturbed input $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x} + \boldsymbol{\epsilon})$ also needs to satisfy the same condition (as informally depicted in Figure 1, right). In other words, Equation (5) needs to hold in a neighborhood of $\boldsymbol{x}$ for robust classification. Thanks to the Lipschitz continuity of V_y and $\boldsymbol{f}_{\boldsymbol{\theta}}$, this can be achieved by a more strict condition than Equation (5) on the dynamics (Theorem 3.1).

Theorem 3.1 (Robust FI Implies Robust Classification)**.** *Consider the dynamical system in Equations (1a) and (1b) with dynamics restricted to the probability simplex $\triangle$ and initial condition $\boldsymbol{\eta}(0) = \mathbb{1}\frac{1}{n}$. If the following conditions hold, then $\boldsymbol{\eta}(T)$ will produce a robust classification of $\boldsymbol{x}$:*

$$\frac{\partial V_y}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) \leq -\kappa, \quad \forall \boldsymbol{\eta} \in \mathcal{D}_y \tag{6}$$

$$\kappa \geq \bar{\epsilon} L_{V_y} L_f^x. \tag{7}$$

where $\bar{\epsilon}$ is the perturbation magnitude on the input, L_{V_y} is the Lipschitz constant of the Lyapunov function, and L_f^x is the Lipschitz constant of the dynamics with respect to x.

See supplementary material B.1 for proof.

3.2. Forward Invariance on a Probability Simplex

For the purposes of certification and training, it is often useful to make the state space $\mathcal{H}$ be a bounded set, as certifying over unbounded sets is typically intractable. For multi-class classification, a natural choice is the probability simplex. Since we initialize $\boldsymbol{\eta}$ within the simplex, it suffices to render the simplex to be forward invariant. We explicitly constrain the states to a probability simplex using a Control-Barrier Function based Quadratic Program (CBF-QP)[2] (Ames et al., 2016), implemented as a differentiable optimization layer (Agrawal et al., 2019).

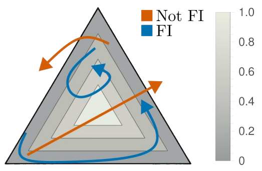

Figure 4. The color contours show level-sets of a barrier function in a 3-class probability simplex.

Barrier functions can be viewed as a variant of Lyapunov functions that only require the state to stay within a set rather than always make progress towards some minimum. Specifically, we choose a potential function h with a 0-super level set (i.e. $\{\boldsymbol{\eta} \in \mathcal{H} | h(\boldsymbol{\eta}) \geq 0\}$) equal to the desired forward invariant set $\mathcal{S}$ (see Figure 4 for an example). Similarly to the Lyapunov case, there is a point-wise inequality condition that must be true over the forward invariant set:

$$\frac{d}{dt} h(\boldsymbol{\eta}(t)) \geq -\alpha(h(\boldsymbol{\eta})) \tag{8}$$

where $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is another class $\mathcal{K}_\infty$ function. Intuitively, all the flows on the boundary of the forward invariant set must have a positive time-derivative (otherwise there could be a point on the boundary that decreases the value of h and thus exits the forward invariant set). This is the essence of Nagumo's theorem (Nagumo, 1942). Barriers extend this idea with a condition that can be applied everywhere in the target forward invariant set without being overly conservative. As trajectories approach the boundary of the set, Equation (8) ensures the time derivative increases until it is positive at the boundary. We use a variation of barrier functions called Control Barrier Functions (CBF). We formalize this concept with Theorem B.2.

In our case, the unconstrained dynamics is the output of a neural network and we denote it as $\hat{f}(\boldsymbol{\eta}, \boldsymbol{x})$. To make the dynamics satisfy the barrier conditions, we use a Control Barrier Function Quadratic Program (CBF-QP) Safety Filter

[2]This is analogous to projected gradient descent (PGD) where we project the dynamics instead of the states.

(Gurriet et al., 2018):

$$f(\hat{\boldsymbol{f}}) = \operatorname*{argmin}_{\mathtt{f}\in\mathbb{R}^n} \frac{1}{2}\|\mathtt{f} - \hat{\boldsymbol{f}}\|_2^2 \tag{9a}$$

$$\text{s.t} \qquad \mathbb{1}^\top \mathtt{f} = 0 \tag{9b}$$

$$\mathtt{f} \geq -\alpha(\boldsymbol{\eta}) \tag{9c}$$

where the arguments to the function $\hat{f}$ are omitted for brevity. Equation (9b) ensures that the sum of the state stays to be 1 and Equation (9c) is the barrier condition that guarantees the state to be non-negative. We provide the detailed explanation in Appendix B.3.

3.3. Certifiable Lyapunov Training

Algorithm 1 Certifiable Lyapunov Training

```
Input      : Lyapunov function $V$, Sampling scheduler, dataset $\mathcal{D}$, perturbation magnitude $\bar{\epsilon}$.
Initialize : Model parameters $\boldsymbol{\theta}$.
for $i$=1:$M$ do
    $(\boldsymbol{x}, y) \sim \mathcal{D}$
    ▷ Sample $\boldsymbol{\eta}$ based on the training progress and the level sets of $V$: $\boldsymbol{\eta} \sim$ Sampling_scheduler $(i, V)$
    ▷ Set $\kappa$ based on the Lipschitz and $\bar{\epsilon}$ (Equation (7))
    ▷ Update model parameters to minimize Lyapunov loss $\mathscr{L}(\boldsymbol{\theta})$ (Equation (10)).
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta\nabla_{\boldsymbol{\theta}}\mathscr{L}(\boldsymbol{\theta})$
return $\boldsymbol{\theta}$
```

Our learning algorithms build upon the LyaNet framework for Lyapunov training (Jimenez Rodriguez et al., 2022), and are summarized in Algorithm 1. The key modifications we make are adaptive sampling to focus on the states that are crucial for certifying forward invariance and controlling the Lipschitz constant of the dynamics.

Training loss. Our training loss is designed to encourage the dynamics to satisfy the conditions in Theorem 3.1. Specifically, we use the Monte Carlo Lyapunov loss in (Jimenez Rodriguez et al., 2022):

$$\mathscr{L}(\boldsymbol{\theta}) \approx \mathop{\mathbb{E}}_{\substack{(\boldsymbol{x},\boldsymbol{y})\sim D \\ \boldsymbol{\eta}\sim\mu(\mathcal{H})}} \left[\max\left\{0, \frac{\partial V_y}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) + \kappa V_y(\boldsymbol{\eta})\right\}\right] \tag{10}$$

Adaptive sampling. To minimize the Lyapunov loss (Equation (10)), we need to sample from a distribution μ. This was simply a uniform distribution in (Jimenez Rodriguez et al., 2022) and may require an intractable number of samples to minimize Equation (10) everywhere in the state space.

We address this challenge with an adaptive sampling strategy that focuses the training samples on regions of the state space that are crucial for satisfying forward invariance: the boundary of the Lyapunov sub-level set. We adapt the sampling distribution as training progresses. We start by sampling uniformly in a hypercube or hypersphere towards the origin as in (Jimenez Rodriguez et al., 2022) since trajectories an untrained NODE can visit anywhere in the state space (see example in Figure 7). As training progresses, we gradually switch to sample within the forward invariant set (the sublevel set of the Lyapunov function), with the switching time as a hyper-parameter defined by the sampling scheduler. This process focuses the sampling on regions that are most important for certifying robust forward invariance.

Restricting Lipschitz constant. Another challenge to obtain a non-vacuous robustness bound is to ensure that κ is larger than a threshold which depends on L_f^x (see the robust forward invariance condition in Equation (7)). In practice, the Lipschitz bound of a small neural network for image classification can be on the order of 10^8 (Huang et al., 2021), and it is unfeasible to train with such a large κ in the Lyapunov loss. Therefore, we need to restrict the Lipschitz constants of the learned dynamics with respect to the input $\boldsymbol{x}$. In addition, since we rely on the smoothness of the dynamics to verify the Lyapunov condition holds on the region of interest (Section 3.4), we also need to restrict the Lipschitz constant with respect to $\boldsymbol{\eta}$ to ensure the dynamics do not change excessively for two states that are close.

To control the Lipschitz constant of our neural network dynamics, we use orthogonal layers (Trockman & Kolter, 2021) and training with a Lipschitz bound in the loss (Tsuzuku et al., 2018). We find that both methods perform similarly, but using orthogonal layers is more computationally efficient and less sensitive to hyper-parameters.

3.4. Certification

In the previous section, we minimize the empirical Lyapunov loss on some finite sample to encourage the dynamics to satisfy conditions in Theorem 3.1. However, zero empirical Lyapunov loss does not guarantee the conditions to hold everywhere on the boundary of the forward invariant set since it only takes finite number of samples. In this section, we develop tools to certify that the forward invariance conditions hold *everywhere* on the boundary of the safe set.

Certification procedure. The certification procedure is as follows: 1) sampling points on the boundary of the forward invariant set (blue dots in Figure 5), and check Lyapunov condition holds on all the sampled points; 2) perturb around each sampled point and verify the condition holds in a small neighborhood around those points.

Sampling techniques (Procedure 1). The key challenge to make the sampling approach rigorous for certification is: the set of samples and their neighborhoods need to cover the whole boundary of the forward invariant set. During training, we do random sampling on the forward invariant set, however, there is no guarantee that the sampled points cover the boundary of the safe set with their neighborhoods. On the other hand, deterministically uniform sampling in a hypercube can cover the whole cube and thus cover the boundary of forward invariant set. However, the sampling efficiency is low since most of the sampled points are not on the boundary. Therefore, we need to develop tools for efficient sampling on the boundary of forward invariant set.

For image classification tasks, the boundary of the Lyapunov level set is the decision boundary in a simplex. In Theorem 3.2, we construct a set, and show that the constructed set can cover the decision boundary.

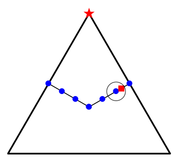
Figure 5. Sampling to cover the decision boundary.

Theorem 3.2 (Sampling on the decision boundary)**.** *For $N \in \mathbb{Z}_+$, $N \equiv 0$ (mod 2) and $N \not\equiv 1$ (mod n), let $S_y = \{\boldsymbol{s} \in \mathbb{Z}^n | \sum_{i=1}^n \boldsymbol{s}_i = N, \boldsymbol{s}_y = \max_{i \neq y} \boldsymbol{s}_i, \boldsymbol{s}_i \geq 0, \forall i = 1, ..., n\}$, and $\tilde{S}_y = \{\tilde{\boldsymbol{s}} \in \mathbb{R}^n | \tilde{\boldsymbol{s}} = \frac{\boldsymbol{s}}{N}, \boldsymbol{s} \in S_y\}$. For any $\boldsymbol{\eta} \in \mathscr{D}_y$, there exists an $\tilde{\boldsymbol{s}} \in \tilde{S}_y$ such that $|\boldsymbol{\eta}_i - \tilde{\boldsymbol{s}}_i| \leq \frac{1}{N}$ for all $i = 1, ..., n$.*

Remark. *The constructed set is sufficient for certification because given any point on the simplex or decision boundary (red dot in Figure 5), there exists a sampled point close by (blue dot), which is shown in Theorem 3.2.*

Verification in a neighborhood around sampled points (Procedure 2). After checking the conditions hold on the sampled points, we need to verify them hold in a small neighborhood around those points. The general idea is to bound the range of values of the output given the range of the input. A direct way of doing so is estimating the Lipschitz constant of the left hand side of Equation (6). By definition of the Lipschitz constant, the norm of output difference can be bounded by the norm of the input difference. Another way is to use linear relaxation based perturbation analysis (LiRPA) (Xu et al., 2020), which is relatively tight and still computationally feasible.

The dynamics of our NODE is parameterized by a neural network followed by a CBF-QP layer. Let $\hat{\boldsymbol{f}}(\boldsymbol{\eta})$ be the dynamics output by the neural network, and let $f(\hat{\boldsymbol{f}})$ be the dynamics after CBF-QP layer. Given perturbed input in an interval bound $\underline{\boldsymbol{\eta}_i} \leq \boldsymbol{\eta}_i \leq \overline{\boldsymbol{\eta}_i}$, we first use a popular linear relaxation based verifier named CROWN (Zhang et al., 2018) to get an interval bound for $\hat{\boldsymbol{f}}$: $\underline{\hat{\boldsymbol{f}}_i} \leq \hat{\boldsymbol{f}}_i \leq \overline{\hat{\boldsymbol{f}}_i}$. However, CROWN does not support perturbation analysis on differentiable optimization layers such as our CBF-QP layer and deriving linear relaxation for CBF-QP can be hard. However, it is possible to derive interval bounds (a special case of linear bounds in CROWN) through CBF-QP. Consider a QP in the form of Equations (9a) to (9c), we can bound each dimension of $f(\hat{\boldsymbol{f}})$ in $\mathcal{O}(n)$ by solving the QP with the corresponding element of the input set to the lower or upper bound. Mathematically, define function h_i to be $h_i : (\boldsymbol{\eta}, \hat{\boldsymbol{f}}) \mapsto f(\hat{\boldsymbol{f}})_i$. Given input interval bound on $\boldsymbol{\eta}$ and $\hat{\boldsymbol{f}}$, we have

$$h_i(\boldsymbol{\eta}_{ub}^i, \hat{\boldsymbol{f}}_{lb}^i) \leq f(\hat{\boldsymbol{f}})_i \leq h_i(\boldsymbol{\eta}_{lb}^i, \hat{\boldsymbol{f}}_{ub}^i) \tag{11}$$

where $\boldsymbol{\eta}_{ub}^i, \boldsymbol{\eta}_{lb}^i$ are defined as follows and $\hat{\boldsymbol{f}}_{ub}^i, \hat{\boldsymbol{f}}_{lb}^i$ are defined in the same way (see Appendix B.5 for proof):

$$\boldsymbol{\eta}_{ub}^i = \begin{cases} \overline{\boldsymbol{\eta}_j}, & j = i \\ \underline{\boldsymbol{\eta}_j}, & j \neq i \end{cases} \quad \boldsymbol{\eta}_{lb}^i = \begin{cases} \underline{\boldsymbol{\eta}_j}, & j = i \\ \overline{\boldsymbol{\eta}_j}, & j \neq i \end{cases} \tag{12}$$

4. Experiments

We evaluate our FI-ODE approach on two applications: 1) certified robustness for image classification (section 4.2), and 2) safety in continuous control (section 4.3). We discuss the computational costs of our method in section 4.4.

4.1. Experimental Setup

For image classification, we first use a neural neural network with 4 convolutional layers and 3 linear layers to encode input image $\boldsymbol{x}$, and then use a 3 layer MLP to model the dynamics in eq. (1b). We evaluate on standard benchmarks, by perturbing the input images within an ℓ_2 ball of radius 0.1 and 0.2 on MNIST, and 0.141 (36/255) on CIFAR-10. For the control problem, we use a 3 layer MLP as the controller. We train and certify that the controller can keep the system to be forward invariant within an ellipsoid around the origin.

4.2. Certified Robustness for Image Classification

We want to evaluate the effectiveness of our FI-ODE approach for training certifiably robust NODEs. To our knowledge, there are no previous works demonstrating certified

Table 1. ℓ_2 certified robustness. Accuracy (%) is reported. Semi-MonDeq results are reported on 100 test images with 95% confidence interval due to high cost, and 10,000 test images are used for other methods.

Dataset	Method	ϵ	Clean	Adversarial	Certified
MNIST	Lipschitz-MonDeq (Pabbaraju et al., 2020)	0.1	95.60	94.42	83.09
	Semi-MonDeq † (Chen et al., 2021)	0.1	99 [>94]	99 [>94]	99 [>94]
	Robust FI-ODE (Ours)	0.1	99.35	99.09	95.75
	Lipschitz-MonDeq (Pabbaraju et al., 2020)	0.2	95.60	93.09	50.56
	Robust FI-ODE (Ours)	0.2	99.35	98.83	81.65
CIFAR-10	Lipschitz-MonDeq (Pabbaraju et al., 2020)	0.141	66.66	50.51	<7.37
	NODE w/o Lyapunov training	0.141	69.05	56.94	16.81
	LyaNet + Lipschitz restriction (Jimenez Rodriguez et al., 2022)	0.141	73.15	64.87	41.43
	LyaNet + Sampling scheduler (Jimenez Rodriguez et al., 2022)	0.141	82.83	74.81	0
	Robust FI-ODE (Ours)	0.141	78.34	67.45	42.27

robustness of NODEs on CIFAR-10. Therefore, we compare with methods on certifying robustness of monotone deep equilibrium models (monDeq) (Pabbaraju et al., 2020; Chen et al., 2021). Similar to NODE, MonDeq (Winston & Kolter, 2020) is also an implicit-depth model.

Table 1 shows the results. The main metric is certified accuracy, the percentage of test set inputs that are certifiably robust. Our approach achieves the strongest overall certified robustness results compared to prior ODE-based approaches. We also reported clean and adversarial accuracy for references. For baseline methods (Pabbaraju et al., 2020; Chen et al., 2021), the adversarial accuracy are evaluated with PGD attack. For our methods, we use AutoAttack (Croce & Hein, 2020) to evaluate the empirical adversarial robustness.

On MNIST, we compare with Lipschitz-MonDeq (Pabbaraju et al., 2020) and Semialgebraic-MonDeq (Chen et al., 2021), and our method significantly outperforms Lipschitz-MonDeq (Pabbaraju et al., 2020). Semialgebraic-MonDeq (Chen et al., 2021) only tested their method on the first 100 test images on MNIST and reported the 95% confidence interval in bracket. Our method performs similarly if not better when comparing accuracy within 95% confidence interval. Since Semialgebraic-MonDeq (Chen et al., 2021) does not scale up to CIFAR-10 (it takes 1350s per sample on MNIST), we only compare with it on MNIST.

On CIFAR-10, Lipschitz-MonDeq (Pabbaraju et al., 2020) achieved 7.37% certified accuracy with $\epsilon = 0.01$. When using the standard benchmark perturbation magnitude 0.141, the certified accuracy should be lower, and our approach significantly outperforms this method. This is potentially because Lipschitz-MonDeq (Pabbaraju et al., 2020) controls the Lipschitz constant of monDeq and restricts the expressiveness of the model. In contrast, our approach only enforces properties on the point-wise dynamics instead of the full trajectories, and thus is less restrictive.

For the rest of the baselines, we train the models differently but certify with the same methods in Section 3.4 to see the effectiveness of the learning algorithms in Section 3.3. All the learning components: Lyapunov training, Lipschitz restriction and adaptive sampling are needed for good performance, and the model that is trained with all of them (Robust FI-ODE) achieved the highest certified accuracy.

First, we train a NODE with standard adjoint method and cross entropy loss instead of Lyapunov training. We still use orthogonal layers (Trockman & Kolter, 2021) to control the Lipschitz constant of the dynamics. The trained model still maintains some certified robustness, mainly because of the restricted Lipschitz constant. Second, we use Lyapunov training (Jimenez Rodriguez et al., 2022) to train the NODE, and test the effectiveness of Lipschitz restriction and adaptive sampling. We see that Lipschitz restriction is crucial for certified robustness. Without Lipschitz restriction, the global Lipschitz bound of the dynamics is on the order of 10^8, making the robustness guarantee vacuous. Comparing with uniform sampling, adaptive sampling focuses on the states that are crucial for certifying robust forward invariance, and making the training to be less restrictive and achieves better clean and adversarial accuracy.

4.3. Certifying Safety for Continuous Control

We test our approach on a planar segway system with three states: angular position ϕ, angular velocity $\dot{\phi}$ and velocity v. Our goal is to train a NODE controller and certify that the controller keeps the states in a forward invariant safe set.

We enforce forward invariance within the c-sublevel set of the Lyapunov function: $\Omega_c(V) = \{x|V(x) \leq c\}$. We learn a quadratic Lyapunov function $V(x) = x^\top Px$, where P is positive definite and learnable, and use adversarial training to increase the penalty for states that tend to violate the Lyapunov condition. Adversarial training is useful to restrict the empirical Lispchitz of the dynamics and eases the certification. During certification, we use rejection sampling to find the states that cover the c-level set of the Lyapunov function (states between the two dashed lines in fig. 6), check that $\dot{V} < 0$ holds on those states, and use CROWN (Raghunathan et al., 2018) to verify the condition also holds around a neighborhood of the sampled states. Then we can show that the set within the solid line is forward invariant.

Figure 6 (a) shows the contours of the time derivative of the learned Lyapunov function on two of the three dimensions when $v = 0$. The solid line is the level set when $V = 0.15$. The two dashed lines show the region where samples are accepted during rejection sampling. Figure 6 (b) shows trajectories with initial states in the forward invariant set (the gray ellipsoid).

4.4. Computational Costs

The main computational costs of our method comes from the number of samples that are needed to cover the boundary of the forward invariant set. Table 2 compares the computational costs and performance for different certification

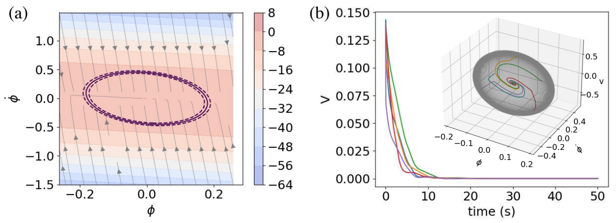

Figure 6. (a) $\dot{V}$ contours and certified forward invariant set (within the solid line). Background gray arrows are the dynamic flows. (b) V along the trajectories. All trajectories starting within the forward invariant set stay in it.

Table 2. Computational costs for certification on CIFAR-10.

Certification Method	Sampling density (N)	# samples	Time (s)	Certified
Lipschitz	20	3.67×10^5	1.03	0
Lipschitz	30	5.50×10^6	1.37	27.40
Lipschitz	40	4.13×10^7	2.8	33.46
CROWN	40	4.13×10^7	240	42.27

methods on CIFAR-10. We first compare the results of certifying with Lipschitz bounds and CROWN (Zhang et al., 2018). Certifying with Lipschitz bounds is faster. Since we can pre-compute the Lipschitz bound of the dynamics, the certification time equals to the inference time on all the states. Certifying with CROWN provides a tighter bound and thus higher certified accuracy but is more computationally expensive than using the Lipschitz bound. We also compare the performance of different sampling density by choosing different N in Theorem 3.2. With larger N, we can cover the region of interest with smaller neighborhood around each sampled point. We vary N using the Lipschitz certification method because it is faster to evaluate, but the pattern should remain the same for CROWN. As expected, we get better accuracy with larger sampling density, but the computational time is longer since we have more samples.

5. Related Works

Empirical Robustness of NODEs. Yan et al. (2020) proposes a steady state loss to improve empirical robustness of NODEs. A line of work trains NODEs to stabilize to Lyapunov equilibrium points. Kang et al. (2021) adds a regularization term in the loss to encourage the Jacobian of dynamics in NODE to have eigenvalues with negative real parts. Huang et al. (2022) paramterizes the weights of NODE blocks to be skew-symmetric. However, both works only demonstrate empirical robustness improvement from NODE but no formal guarantees. Huang et al. (2022) also shows that the empircal robustnesss improvement may be due to the gradient masking effects from the adaptive ODE solvers.

Verification and Certified robustness of NODEs. Previous works on formal analysis of NODEs mostly focus on reachability analysis. Stochastic Lagrangian Reachability (SLR) (Grunbacher et al., 2021) proposes an abstraction-based technique to provide stochastic bound on the reachable set of NODEs. Lopez et al. (2022) computes deterministic reachable set of NODEs via zonotope and polynomial-zonotope based methods implemented in CORA (Althoff, 2013). However, these methods have only been demonstrated on very low dimension problems or on linear NODEs. Another important class of implicit-depth models is monDEQ (Winston & Kolter, 2020), and it can be thought of as an implicit ODE. There are previous works trying to certify ℓ_2 robustness of monDEQ, either via controlling Lipschitz (Pabbaraju et al., 2020) or semialgebraic representation (Chen et al., 2021). However, they do not scale well beyond MNIST. Recently, Xiao et al. (2022) propose invariance propagation for stacked NODEs that provides guarantees for output specifications by controller/input synthesis. While their approach focuses more on interpretable causal reasoning of stacked NODEs, our work provides a framework for training and provably certifying general NODEs.

Formal verification of neural networks. Formal verification of neural networks aim to prove or disprove certain specifications of neural networks, and a canonical problem of neural network verification is to bound the output of neural networks given specified input perturbations. Computing the exact bounds is a NP-complete problem (Katz et al., 2017) and can be solved via MIP or SMT solvers (Tjeng et al., 2019; Ehlers, 2017), but they are not scalable and often too expensive for practical usage. In the meanwhile, incomplete neural network verifiers are developed to give sound outer bounds of neural networks (Salman et al., 2019; Dvijotham et al., 2018; Wang et al., 2018; Singh et al., 2019), and bound-propagation-based methods such as CROWN (Zhang et al., 2018) are a popular approach for incomplete verification. Recently, branch-and-bound based approaches (Bunel et al., 2020; Wang et al., 2021; De Palma et al., 2021) are proposed to further enhance the strength of neural network verifiers. Our work utilizes neural network verifiers as a sub-procedure to prove forward invariance of NODEs, and is agnostic to the verification algorithm used. We used CROWN because it is efficient, GPU-accelerated and has high quality implementation (Xu et al., 2020).

Learning Lyapunov functions and controllers for nonlinear control problems. A range of work learns neural network Lyapunov functions (Chang et al., 2019), barrier functions (Jin et al., 2020), and control policies (Dai et al., 2021) for nonlinear control problems. To certify the stability or safety requirements, Chang et al. (2019) uses SMT solvers (Gao et al., 2013), Jin et al. (2020) uses Lispchitz methods and (Dai et al., 2021) formulate the verification as mixed integer programming (MIP). We use a linear relaxation based neural network verifier (Zhang et al., 2018) that enjoys a good balance between tightness and computational cost, enabling us to certify nonlinear control policies (Chang et al. (2019); Jin et al. (2020) learns linear control policies) on the real dynamics (Dai et al. (2021) uses a neural network to approximate the dynamics).

6. Conclusion

We have presented FI-ODE: a framework for certified and robust forward invariance of NODEs in both the classification and control contexts. For image classification, we certify adversarial robustness by enforcing robust forward invariance of sublevel sets of Lyapunov functions. We demonstrate the broad applicability of our FI-ODE by certifying the forward invariance of a region of the state-space in a planar segway model. We consider this a first step towards certification of desirable properties in complex NODEs for a wide variety of domains from classification to classical control.

Acknowledgements. This work is funded in part by AeroVironment and NSF #1918865.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Althoff, M. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th international conference on hybrid systems: computation and control*, pp. 173–182, 2013.

Ames, A. D., Galloway, K., Sreenath, K., and Grizzle, J. W. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 2014.

Ames, A. D., Xu, X., Grizzle, J. W., and Tabuada, P. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 2016.

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

Bunel, R., Lu, J., Turkaslan, I., Kohli, P., Torr, P., and Mudigonda, P. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research (JMLR)*, 2020.

Chang, Y.-C., Roohi, N., and Gao, S. Neural lyapunov control. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Chen, T., Lasserre, J. B., Magron, V., and Pauwels, E. Semialgebraic representation of monotone deep equilibrium models and applications to certification. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 27146–27159, 2021.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning (ICML)*, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, pp. 2206–2216. PMLR, 2020.

Dai, H., Landry, B., Yang, L., Pavone, M., and Tedrake, R. Lyapunov-stable neural-network control. *Robotics: Science and Systems (RSS)*, 2021.

De Palma, A., Behl, H. S., Bunel, R., Torr, P. H. S., and Kumar, M. P. Scaling the convex barrier with active sets. *International Conference on Learning Representations (ICLR)*, 2021.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P. A dual approach to scalable verification of deep networks. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

E, W. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5 (1):1–11, 2017.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, 2017.

Gao, S., Kong, S., and Clarke, E. M. dreal: An smt solver for nonlinear theories over the reals. In *International conference on automated deduction*, pp. 208–214. Springer, 2013.

Grunbacher, S., Hasani, R., Lechner, M., Cyranka, J., Smolka, S. A., and Grosu, R. On the verification of neural odes with stochastic guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Gurriet, T., Singletary, A., Reher, J., Ciarletta, L., Feron, E., and Ames, A. Towards a framework for realizable safety critical control through active set invariance. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, dec 2017.

Huang, Y., Zhang, H., Shi, Y., Kolter, J. Z., and Anandkumar, A. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Huang, Y., Yu, Y., Zhang, H., Ma, Y., and Yao, Y. Adversarial robustness of stabilized neural ode might be from obfuscated gradients. In *Mathematical and Scientific Machine Learning*. PMLR, 2022.

Jimenez Rodriguez, I. D., Ames, A. D., and Yue, Y. Lyanet: A lyapunov framework for training neural odes. In *International Conference on Machine Learning (ICML)*, 2022.

Jin, W., Wang, Z., Yang, Z., and Mou, S. Neural certificates for safe control policies. *arXiv preprint arXiv:2006.08465*, 2020.

Kang, Q., Song, Y., Ding, Q., and Tay, W. P. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification (CAV)*, 2017.

Khalil, H. K. Nonlinear systems third edition. *Patience Hall*, 115, 2002.

Khalil, I., Doyle, J., and Glover, K. *Robust and optimal control*. prentice hall, new jersey, 1996.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. How does noise help robustness? explanation and exploration under the neural sde framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 282–290, 2020.

Lopez, D. M., Musau, P., Hamilton, N., and Johnson, T. T. Reachability analysis of a general class of neural ordinary differential equations. *arXiv preprint arXiv:2207.06531*, 2022.

Nagumo, M. Über die lage der integralkurven gewöhnlicher differentialgleichungen. *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, 1942.

Pabbaraju, C., Winston, E., and Kolter, J. Z. Estimating lipschitz constants of monotone deep equilibrium models. In *International Conference on Learning Representations (ICLR)*, 2020.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages (POPL)*, 2019.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. *International Conference on Learning Representations (ICLR)*, 2019.

Trockman, A. and Kolter, J. Z. Orthogonalizing convolutional layers with the cayley transform. *International Conference on Learning Representations (ICLR)*, 2021.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Wang, S., Pei, K., Whitehouse, J., Yang, J., and Jana, S. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Winston, E. and Kolter, J. Z. Monotone operator equilibrium networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:10718–10728, 2020.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, 2018a.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML)*, 2018b.

Xiao, W., Wang, T.-H., Hasani, R., Lechner, M., and Rus, D. On the forward invariance of neural odes. *arXiv preprint arXiv:2210.04763*, 2022.

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Xu, X., Tabuada, P., Grizzle, J. W., and Ames, A. D. Robustness of control barrier functions for safety critical control. *IFAC-PapersOnLine*, 2015.

Yan, H., Du, J., Tan, V. Y., and Feng, J. On robustness of neural ordinary differential equations. *International Conference on Learning Representations (ICLR)*, 2020.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

A. Definitions

A.1. Class $\mathcal{K}$ functions

Definition 3 (Class $\mathcal{K}$ Function). *A continuous function $\alpha : [0, a) \to [0, \infty)$ for $a \in \mathbb{R}_{>0} \cup \{\infty\}$ belongs to class $\mathcal{K}$ ($a \in \mathcal{K}$) if it satisfies:*

1. ***Zero at Zero:*** $\alpha(0) = 0$
2. ***Strictly Increasing:*** *For all $r_1, r_2 \in [0, a]$ we have that $r_1 < r_2 \Rightarrow \alpha(r_1) < \alpha(r_2)$*

Definition 4 (Class $\mathcal{K}_\infty$ Function). *A function belongs to $\mathcal{K}_\infty$ if it satisfies:*

1. $\alpha \in \mathcal{K}$
2. ***Radially Unbounded:*** $\lim_{r\to\infty} \alpha(r) = \infty$

Definition 5 (Extended Class $\mathcal{K}^e_\infty$ Function). *A continuous function $\alpha : \mathbb{R} \to \mathbb{R}$ belongs to extended $\mathcal{K}^e_\infty$ if it satisfies:*

1. ***Zero at Zero:*** $\alpha(0) = 0$
2. ***Strictly Increasing:*** *For all $r_1, r_2 \in [0, a]$ we have that $r_1 < r_2 \Rightarrow \alpha(r_1) < \alpha(r_2)$*

Definition 6 (Class $\mathcal{KL}$ Function). *A continuous function $\beta : [0, a) \times [0, \infty) \to [0, \infty)$ belongs to $\mathcal{KL}$ if it satisfies:*

1. ***Class $\mathcal{K}$ on first argument:*** $\forall s \in [0, \infty) \beta(\cdot, s) \in \mathcal{K}$
2. ***Asymptotically 0 on second argument:*** $\forall r \in [0, a) \lim_{s\to\infty} \beta(r, s) = 0$

A.2. Barrier Loss

Recall that $\boldsymbol{\eta} \in \mathcal{H} \subset \mathbb{R}^k$. Suppose we wish to render the set $\mathcal{S} = \{\boldsymbol{\eta} \in \mathcal{H} : h(\boldsymbol{\eta}) \geq 0\}$ where $h : \mathcal{H} \to \mathbb{R}$ is a continuously differentiable function. Systems that satisfy barrier conditions in Theorem B.2 provides two properties: forward invariance of $\mathcal{S}$ and stability of the system towards $\mathcal{S}$. This stability can be local to a subset of the state-space that contains the desired set. Namely, we expect the property $\mathcal{S} \subseteq \mathcal{R} \subseteq \mathcal{H}$ where $\mathcal{R}$ is the region of attraction. With this added detail to the problem setting we are ready to state the definition of Barrier loss:

Definition 7 (Barrier Loss). *For a barrier function candidate $h : \mathcal{H} \to \mathbb{R}$, with function $\alpha \in \mathcal{K}^e_\infty$ and its corresponding region of attraction $\mathcal{R}$, the Barrier Loss $\mathscr{L} : \boldsymbol{\Theta} \to \mathbb{R}_{\geq 0}$ is defined as:*

$$\mathscr{L}(\boldsymbol{\theta}) = \int_{\mathcal{R}\times[0,1]} \max\{0, -\frac{\partial h}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}, t) - \alpha(h(\boldsymbol{\eta}))\} d\boldsymbol{\eta} \times t \tag{13}$$

B. Theorems with Proof

Theorem B.1 (Exponentially Stabilizing Control Lyapunov Function (ES-CLF) Implies Exponential Stability (Ames et al., 2014))**.** *For the ODE in eqs. (1a) and (1b), a continuously differentiable function $V : \mathbb{R}^k \to \mathbb{R}_{\geq 0}$ is an ES-CLF if there are class $\mathcal{K}_\infty$ functions $\overline{\sigma}$ and κ such that:*

$$V(\boldsymbol{\eta}) \leq \overline{\sigma}(\|\boldsymbol{\eta}\|), \tag{14}$$

$$\min_{\boldsymbol{\theta}\in\Theta} \left[\frac{\partial V}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) + \kappa(V(\boldsymbol{\eta})) \right] \leq 0 \tag{15}$$

holds for all $\boldsymbol{\eta} \in \mathcal{R} \subseteq H$ and $t \in [0, 1]$. The existence of an ES-CLF implies that there is a $\boldsymbol{\theta} \in \Theta$ that can achieve:

$$\frac{\partial V}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) + \kappa(V(\boldsymbol{\eta})) \leq 0, \tag{16}$$

and furthermore the ODE using $\boldsymbol{\theta}$ is exponentially stable with respect to V, i.e., $V(\boldsymbol{\eta}(t)) \leq V(\boldsymbol{\eta}(0))e^{-\underline{\kappa}t}$ for some $\underline{\kappa} > 0$.

Proof. Since Θ is compact, minimums are attained within Θ. Let $\boldsymbol{\theta}^*(\boldsymbol{\eta}) = \mathrm{argmin}_{\boldsymbol{\theta}\in\Theta} \frac{\partial V}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x})$. Therefore, from eq. (15) we can conclude that:

$$\frac{\partial V}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}^*(\boldsymbol{\eta})}(\boldsymbol{\eta}, \boldsymbol{x}) \leq -\kappa(V(\boldsymbol{\eta})) \qquad \forall \boldsymbol{\eta} \in \mathcal{H} \tag{17}$$

For simplicity we will omit the arguments of $\boldsymbol{\theta}^*$. Furthermore, in the case where $\boldsymbol{\theta}^*$ is a set we will select only one. Since $\mathcal{H}$ is compact then

$$\sigma^* = \max_{\boldsymbol{\eta}\in\mathcal{H}} \overline{\sigma}(\|\boldsymbol{\eta}\|). \tag{18}$$

This in turn implies that V in bounded in $\mathcal{H}$ which helps us conclude that

$$\overline{V} = \max_{\boldsymbol{\eta}\in\mathcal{H}} V(\boldsymbol{\eta}) \tag{19}$$

is well defined which in turn implies

$$\underline{\kappa} = \min_{r\in[0,\overline{V}]} \frac{d\kappa(r)}{dr} \tag{20}$$

is well defined. Since κ is strictly increasing, then $\underline{\kappa} > 0$. Notice that $\alpha(r) = \underline{\kappa} r$ satisfies $\alpha \in \mathcal{K}_\infty$ and, by the comparison lemma, $\forall r, \underline{\kappa} r \leq \kappa(r)$. Therefore:

$$\frac{\partial V}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}^*(\boldsymbol{\eta})}(\boldsymbol{\eta}, \boldsymbol{x}) \leq -\kappa(V(\boldsymbol{\eta})) \leq -\underline{\kappa} V(\boldsymbol{\eta}), \qquad \forall \boldsymbol{\eta} \in \mathcal{H} \tag{21}$$

In preparation for applying the comparison lemma we will consider the following Initial Value Problem (IVP):

$$y(0) = V(\boldsymbol{\eta}_0) \tag{22}$$
$$\dot{y} = -\underline{\kappa} y \tag{23}$$

Since this is a linear system solutions for y exist, are unique and take the form $y(t) = V(\boldsymbol{\eta}_0)e^{-\underline{\kappa}t}$. Furthermore, by the comparison lemma we can conclude that:

$$V(\boldsymbol{\eta}(t)) \leq y(t) = V(\boldsymbol{\eta}_0)e^{-\underline{\kappa}t} \tag{24}$$

□

Lemma 1 (Solution of Class $\mathcal{K}$ function systems (See Lemma 4.4 in (Khalil, 2002))). *Let $\alpha \in \mathcal{K}^e_\infty$. Then consider the following IVP for $t \in [0, 1]$:*

$$y(0) = y_0 \tag{25}$$
$$\dot{y} = -\alpha(y) \tag{26}$$

This IVP has unique solutions $y(t) = \beta(y_0, t)$ where $\beta \in \mathcal{KL}$.

Theorem B.2 (Control Barrier Function (CBF) Implies Forward Invariance (Xu et al., 2015; Nagumo, 1942))**.** *Let the set $\mathcal{S} \subset \mathcal{H}$ be the 0 superlevel set of a continuously differentiable function $h : \mathcal{H} \to \mathbb{R}$, i.e. $\mathcal{S} = \{\boldsymbol{\eta} \in \mathcal{H} | h(\boldsymbol{\eta}) \geq 0\}$. The set $\mathcal{S}$ is forward invariant with respect to the ODE eqs.* (1a) *and* (1b)*, if h satisfies either of the following conditions:*

1. *(Xu et al., 2015) There exists a function α for all $\boldsymbol{\eta} \in \mathcal{S}$ so that:*

$$\max_{\boldsymbol{\theta}\in\Theta} \left[\frac{\partial h}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) + \alpha(h(\boldsymbol{\eta})) \right] \geq 0, \tag{27}$$

where α is a class $\mathcal{K}^e_\infty$ function (this means $\alpha : \mathbb{R} \to \mathbb{R}$ is strictly increasing and satisfies $\lim_{r\to\infty} \alpha(r) = \infty$).

2. *(Nagumo, 1942) For all $\boldsymbol{\eta} \in \{\boldsymbol{\eta} \in \mathcal{S} | h(\boldsymbol{\eta}) = 0\}$:*

$$\frac{\partial h}{\partial \boldsymbol{\eta}} \neq \mathbf{0} \tag{28a}$$

$$\max_{\boldsymbol{\theta} \in \Theta} \left[\frac{\partial h}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) \right] \geq 0. \tag{28b}$$

Proof. Both conditions follow from fundamentally different arguments. Condition 1 follows from the comparison lemma. Condition 2 uses Nagumo's theorem. In either case we rely on the compactness of Θ to solve the following optimization problem:

$$\boldsymbol{\theta}^*(\boldsymbol{\eta}) = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}} \left[\frac{\partial h}{\partial \boldsymbol{\eta}}^\top \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) \right] \tag{29}$$

We will omit the parameters of $\boldsymbol{\theta}^*$ for brevity and choose a random solution in the case where $\boldsymbol{\theta}^*$ returns a set of solutions.

1. Consider the following IVP:

$$y(0) = h(\boldsymbol{\eta}(0)) \tag{30}$$

$$\dot{y} = -\alpha(y) \tag{31}$$

   which satisfies the conditions of lemma 1. This implies that $y(t)$ is unique and $y(t) = \beta(h(\boldsymbol{\eta}(0)), t)$ where by the assumption of forward invariance $h(\boldsymbol{\eta}(0)) \geq 0$. The by a trivial variant of the comparison lemma we have that $\dot{h}(\boldsymbol{\eta}(t)) \geq -\alpha(h(\boldsymbol{\eta}))$ implies $h(\boldsymbol{\eta}(t)) \geq \beta(h(\boldsymbol{\eta}(0)), t)$ which implies $h(\boldsymbol{\eta}(t)) \geq 0$ for $t \in [0, 1]$

2. This is a direct application of Nagumo's theorem (Nagumo, 1942).

□

B.1. Proof of Theorem 3.1

Proof. Define barrier function as $h = \underline{V} - V$, then the correct classification region $\mathcal{S}_y = \{\boldsymbol{\eta} | \boldsymbol{\eta} \in \triangle, y = \operatorname{argmax} \boldsymbol{\eta}\}$ is a 0-superlevel set of h. According to Nagumo's theorem (Nagumo, 1942), if $\dot{h}(\boldsymbol{x} + \boldsymbol{\epsilon}; \boldsymbol{\eta}) \geq 0$ on $\partial \mathcal{S}_y = \mathcal{D}_y = \{\boldsymbol{\eta} \in \triangle | V(\boldsymbol{\eta}) = 1\}$, then $\mathcal{S}$ is forward invariant. Since $\dot{V}(\boldsymbol{\eta}; \boldsymbol{x}) \leq -\kappa$ on $\mathcal{D}_y$, we have $\dot{h}(\boldsymbol{\eta}; \boldsymbol{x}) \geq \kappa \underline{V}$ on $\partial \mathcal{S}$. Then for perturbed input, we have

$$\begin{aligned}
\dot{h}(\boldsymbol{\eta}; \boldsymbol{x} + \boldsymbol{\epsilon}) &= \dot{h}(\boldsymbol{\eta}; \boldsymbol{x}) + \dot{h}(\boldsymbol{\eta}; \boldsymbol{x} + \boldsymbol{\epsilon}) - \dot{h}(\boldsymbol{\eta}; \boldsymbol{x}) && (32)\\
&\geq \dot{h}(\boldsymbol{\eta}; \boldsymbol{x}) - \|\dot{h}(\boldsymbol{\eta}; \boldsymbol{x} + \boldsymbol{\epsilon}) - \dot{h}(\boldsymbol{\eta}; \boldsymbol{x})\| && (33)\\
&\geq \dot{h}(\boldsymbol{\eta}; \boldsymbol{x}) - L_h L_f^x \bar{\epsilon} && (34)\\
&\geq \kappa \underline{V} - L_h L_f^x \bar{\epsilon} \geq 0 && (35)
\end{aligned}$$

where L_h is the Lipschitz constant of h, L_f^x is the Lipschitz constant of f with respect to x, and $\bar{\epsilon}$ is the norm of $\boldsymbol{\epsilon}$. □

B.2. Proof of Theorem 3.2

Proof. For any $\boldsymbol{\eta} \in \mathscr{D}_y$, let $\boldsymbol{z} = [N\boldsymbol{\eta}_1, ..., N\boldsymbol{\eta}_n]$. By definition of $\mathscr{D}_y$, in addition to $\sum_i \boldsymbol{z}_i = N$, we have

$$\sum_i \boldsymbol{z}_i = N \tag{36}$$

$$\boldsymbol{z}_y = \max_{j \neq y} \boldsymbol{z}_j \tag{37}$$

Define $\tilde{\boldsymbol{z}} = [\boldsymbol{z}_1 - \lfloor \boldsymbol{z}_1 \rfloor, ..., \boldsymbol{z}_n - \lfloor \boldsymbol{z}_n \rfloor]$ to be the vector that contains the fractional part of each element in $\boldsymbol{z}$. Then we sort $\tilde{\boldsymbol{z}}$ in a non-decreasing order. For the tied elements that equals to $\tilde{\boldsymbol{z}}_y$, we put $\tilde{\boldsymbol{z}}_y$ as the last. We denote the sorted vector as

$\tilde{\boldsymbol{z}}' = [\tilde{\boldsymbol{z}}_{i_1}, ..., \tilde{\boldsymbol{z}}_{i_n}]$, where $\tilde{\boldsymbol{z}}_{i_1} \leq ... \leq \tilde{\boldsymbol{z}}_{i_n}$. Let $v : \mathbb{Z}^+ \to \mathbb{Z}^+$ to be a function that maps the indices in $\tilde{\boldsymbol{z}}$ to the indices in $\tilde{\boldsymbol{z}}'$. For instance, if $\tilde{\boldsymbol{z}}_1$ becomes the third element in $\tilde{\boldsymbol{z}}'$, then $v(1) = 3$. If $\tilde{\boldsymbol{z}}_j = \tilde{\boldsymbol{z}}_y$, we have $v(y) > v(j)$.

Notice that $\sum_i \tilde{\boldsymbol{z}}'_i = \sum_i \tilde{\boldsymbol{z}}_i = \sum_i \boldsymbol{z}_i - \sum_i \lfloor \boldsymbol{z}_i \rfloor = N - \sum_i \lfloor \boldsymbol{z}_i \rfloor$, and $\sum_i \tilde{\boldsymbol{z}}'_i < n$ since $0 \leq \tilde{\boldsymbol{z}}'_i < 1$ for $i = 1, ..., n$. Let $k = n - (N - \sum_i \lfloor \boldsymbol{z}_i \rfloor)$, we have

$$\tilde{\boldsymbol{z}}'_1 + ... + \tilde{\boldsymbol{z}}'_k = (1 - \tilde{\boldsymbol{z}}'_{k+1}) + ... + (1 - \tilde{\boldsymbol{z}}'_n) \tag{38}$$

Define vector $\boldsymbol{q}$ as follows:

$$\boldsymbol{q}_{i_j} = \begin{cases} \lfloor N\boldsymbol{\eta}_{i_j} \rfloor, & j = 1, ..., k \\ \lceil N\boldsymbol{\eta}_{i_j} \rceil, & j = k+1, ..., n \end{cases} \tag{39}$$

Then we have $|\boldsymbol{q}_i - \boldsymbol{z}_i| < 1$ for all $i = 1, ..., n$. Now we check $\boldsymbol{q}$ satisfies Equation (36) and a relaxed version of Equation (37). First, we have $\sum_i \boldsymbol{q}_i = N$ because of Equation (38). Next, we show $\boldsymbol{q}_y \geq \max_{i \neq y} \boldsymbol{q}_i$ by contradiction. Suppose there exists an index j such that $\boldsymbol{q}_j > \boldsymbol{q}_y$, then it has to be the case where $\lfloor \boldsymbol{z}_j \rfloor = \lfloor \boldsymbol{z}_y \rfloor$ and we take ceiling on $\boldsymbol{z}_j$ and take floor on $\boldsymbol{z}_y$, i.e. $v(j) > k$ and $v(y) \leq k$. This means $\tilde{\boldsymbol{z}}_j > \tilde{\boldsymbol{z}}_y$, because v is the sorted indices of $\tilde{\boldsymbol{z}}$ in a non-decreasing order and this gives $\tilde{\boldsymbol{z}}_j \geq \tilde{\boldsymbol{z}}_y$, and if $\tilde{\boldsymbol{z}}_j = \tilde{\boldsymbol{z}}_y$, we have $v(y) > v(j)$, which is contradictory to $v(j) > k$ and $v(y) \leq k$. Then we have $\boldsymbol{z}_y = \lfloor \boldsymbol{z}_y \rfloor + \tilde{\boldsymbol{z}}_y < \boldsymbol{z}_j = \lfloor \boldsymbol{z}_j \rfloor + \tilde{\boldsymbol{z}}_j$, which is contradictory to Equation (37). Therefore, there does not exist a j such that $\boldsymbol{q}_j > \boldsymbol{q}_y$, i.e. $\boldsymbol{q}_y \geq \max_{i \neq y} \boldsymbol{q}_i$. For the cases where $\boldsymbol{q}_y = \max_{i \neq y} \boldsymbol{q}_i$, we have $\boldsymbol{q} \in S_y$, i.e. $\boldsymbol{q}$ is a sampled point.

For the cases where $\boldsymbol{q}_y > \max_{i \neq y} \boldsymbol{q}_i$, we show that we can modify $\boldsymbol{q}$ to $\tilde{\boldsymbol{q}}$ such that $\tilde{\boldsymbol{q}} \in S_y$ and $|\tilde{\boldsymbol{q}}_i - \boldsymbol{z}_i| \leq 1$ for all $i = 1, ..., n$. Let $\mathcal{I} = \{i \in \mathbb{Z}^+ | \boldsymbol{z} \neq y, \boldsymbol{z}_i = \boldsymbol{z}_y\}$ be the set that contains the indices of all runner-up elements in $\boldsymbol{z}$. If $\boldsymbol{q}_y > \max_{i \neq y} \boldsymbol{q}_i$, then we must have $\boldsymbol{q}_y = \lceil N\boldsymbol{\eta}_y \rceil$, and $\boldsymbol{q}_i = \lfloor N\boldsymbol{\eta}_i \rfloor$ for all $i \in \mathcal{I}$. We first let $\tilde{\boldsymbol{q}} = \boldsymbol{q}$, and then pick an i^* from $\mathcal{I}$. Let $\mathcal{J} = \{j \in \mathbb{Z}^+ | \boldsymbol{q}_j \geq 1, j \neq i^*, j \neq y\}$. Notice that $\boldsymbol{q}_{i^*} + \boldsymbol{q}_y = 2\lfloor \boldsymbol{z}_y \rfloor + 1$, which is an odd number. Since $\sum_i \boldsymbol{q}_i = N$ and N is an even number, $\mathcal{J} \neq \emptyset$. We discuss how to obtain $\tilde{\boldsymbol{q}}$ case by case.

Case 1: If there exists a $j \in \mathcal{J}$ such that $v(j) > k$, we set $\tilde{\boldsymbol{q}}_j = \lfloor N\boldsymbol{\eta}_j \rfloor$, and set $\tilde{\boldsymbol{q}}_{i^*} = \lceil N\boldsymbol{\eta}_{i^*} \rceil$. Then we have $\sum_i \tilde{\boldsymbol{q}} = \sum_{i \neq i^*, i \neq j} \boldsymbol{q}_i + \boldsymbol{q}_{i^*} + 1 + \boldsymbol{q}_j - 1 = N$ and $|\tilde{\boldsymbol{q}}_i - \boldsymbol{z}_i| \leq 1$ for all $i = 1, ..., n$.

Case 2: If $v(j) \leq k$ for all $j \in \mathcal{J}$, there must exist a j such that $\boldsymbol{q}_j < \boldsymbol{q}_{i^*}$. Otherwise, $\boldsymbol{q}_y = \boldsymbol{q}_{i^*} + 1$, and $\boldsymbol{q}_i = \boldsymbol{q}_{i^*}$ for all $i \neq y$. Then $\sum_i \boldsymbol{q}_i = N = n\boldsymbol{q}_{i^*} + 1$, which is contradictory to the assumption that $N \not\equiv 1 (\text{mod } n)$. Then set $\tilde{\boldsymbol{q}}_j = \lceil N\boldsymbol{\eta}_j \rceil$ and $\tilde{\boldsymbol{q}}_y = \lfloor N\boldsymbol{\eta}_y \rfloor$. Since $\boldsymbol{q}_j < \boldsymbol{q}_{i^*}$, we have $\tilde{\boldsymbol{q}}_j = \boldsymbol{q}_j + 1 \leq \tilde{\boldsymbol{q}}_y = \boldsymbol{q}_{i^*}$. □

Remark. *The assumption that $N \not\equiv 1 (mod\ n)$ is easy to satisfy. Since we also require N is an even number, as long as n is also an even number, we have $N \not\equiv 1 (mod\ n)$. We can also relax this assumption by adding $[\frac{N}{n}, ..., \frac{N}{n}]$ to S_y.*

B.3. Forward Invariance on a Probability Simplex via CBF-QP

Recall that an n-class probability simplex is defined as $\triangle = \{\boldsymbol{\eta} \in \mathbb{R}^n | \sum_{i=1}^n \boldsymbol{\eta}_i = 1, \boldsymbol{\eta}_i \geq 0\}$. Now we show that Equation (9b) ensures that the sum of the state stays to be 1 and Equation (9c) guarantees the state to be non-negative.

First, we need the sum of $\boldsymbol{\eta}$ stays the same as the initial condition. Taking time derivative of both sides of $\sum_{i=1}^n \boldsymbol{\eta}_i = 1$, we have $\frac{d}{dt} \left(\sum_{i=1}^n \boldsymbol{\eta}_i(t)\right) = \sum_{i=1}^n f_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x})_i = \mathbb{1}^\top f_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) = 0$, which is Equation (9b). This is natural because the dynamics summing up to zero means the changes from all dimensions summing up to zero, and thus the sum of all dimensions stays the same.

Next, we need each dimension of the state to be non-negative. Since the initial condition has non-negative entries, we just need the set $\{\boldsymbol{\eta} | \boldsymbol{\eta} \geq 0\}$ to be forward invariant. We define forward invariance via barrier functions. For each dimension i, we define $h_i(\boldsymbol{\eta}) = \boldsymbol{\eta}_i$. Then the 0-superlevel set of h_i equals the safe set $\{\boldsymbol{\eta} | \boldsymbol{\eta}_i \geq 0\}$. As long as the condition in Equation (8) holds, i.e. $\frac{dh_i}{d\boldsymbol{\eta}} f_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) \geq -\alpha(h_i(\boldsymbol{\eta}))$ for some class $\mathcal{K}_\infty$ function α, the set $\{\boldsymbol{\eta} | \boldsymbol{\eta}_i \geq 0\}$ is forward invariant. Plugging in $h_i(\boldsymbol{\eta})$, we have $f_{\boldsymbol{\theta}}(\boldsymbol{\eta}, \boldsymbol{x}) \geq -\alpha(\boldsymbol{\eta})$, which is Equation (9c).

To learn in this setting we differentiate through the QP layer using the KKT conditions as shown in (Agrawal et al., 2019). Given the simple nature of the QP, we implemented a custom solver that uses binary search to efficiently compute solutions, detailed in the supplymentary materials.

To demonstrate the effectiveness of the CBF-QP layer, we visualize the learned trajectories on the CIFAR-3 dataset (a subset of CIFAR-10 with the first 3 classes) in Figure 7. Each colored line represents a trajectory of an input image from a specific class. As training progresses, the trajectories are trained to evolve to the correct classes. All the trajectories stay in the simplex, implying that the learned dynamics satisfy the constraints in Equation (9b) and Equation (9c).

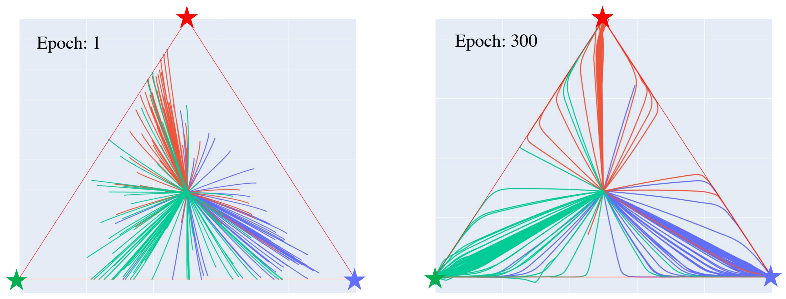

Figure 7. Depicting ODE trajectories that satisfy the simplex constraint for CIFAR-3 on epochs 1 and 300. Each colored line represents the trajectory of an input example of a specific class, and the stars at the corners are colored with the ground-truth class.

B.4. Custom solver for the CBF-QP

Consider a CBF-QP in the following form:

$$\begin{aligned} f(\hat{\boldsymbol{f}}) = \underset{\mathtt{f}\in\mathbb{R}^n}{\operatorname{argmin}} \ & \frac{1}{2}\|\mathtt{f}-\hat{\boldsymbol{f}}\|_2^2 \\ \text{s.t} \quad & \mathbb{1}^\top \mathtt{f} = b \\ & \underline{\mathtt{f}}(\boldsymbol{\eta}) \le \mathtt{f} \le \overline{\mathtt{f}}(\boldsymbol{\eta}) \end{aligned} \tag{40}$$

where $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$ are non-increasing function of $\boldsymbol{\eta}$. By the Karush–Kuhn–Tucker (KKT) conditions, the solution of (40) is as follows:

$$f(\hat{\boldsymbol{f}}) = \left[\hat{\boldsymbol{f}} + \lambda^* \mathbb{1}\right]_{\underline{\mathtt{f}}}^{\overline{\mathtt{f}}} \tag{41}$$

where $[\cdot]_{\underline{\mathtt{f}}}^{\overline{\mathtt{f}}}$ stands for lower and upper clipping by $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$, and λ^* is the Lagrangian multiplier. We find λ^* such that $\mathbb{1}^\top f(\hat{\boldsymbol{f}}) = b$ using binary search. Since $f(\hat{\boldsymbol{f}})$ is clipped by $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$, the search range of λ^* is $[\min_i(\hat{\boldsymbol{f}}_i - \underline{\mathtt{f}}_i), \max_i(\overline{\mathtt{f}}_i - \hat{\boldsymbol{f}}_i)]$, where $\hat{\boldsymbol{f}}_i$ stands for the ith element in $\hat{\boldsymbol{f}}$, and $\underline{\mathtt{f}}_i$, $\overline{\mathtt{f}}_i$ stand for the ith element in $\underline{\mathtt{f}}(\boldsymbol{\eta})$ and $\overline{\mathtt{f}}(\boldsymbol{\eta})$ respectively. Here we consider a general constraint where there are both lower and upper bounds on $\mathtt{f}$. If there is only a lower bound constraint on $\mathtt{f}$ as in eq. (9c), we search λ^* in $[\min_i(\hat{\boldsymbol{f}}_i - \underline{\mathtt{f}}_i), -\min_i \hat{\boldsymbol{f}}_i]$, because if $\lambda^* > -\min_i \hat{\boldsymbol{f}}_i$, then $\mathbb{1}^\top \mathtt{f} > 0$, violating eq. (9b).

To differentiate through the solver in training, we derive the derivatives based on the binding conditions of the inequality constraints. First, we define the binding and not binding sets as follows:

$$\mathcal{S} = \{i | f_i = \underline{\mathtt{f}}_i \text{ or } f_i = \overline{\mathtt{f}}_i\}, \quad \mathcal{S}^c = \Omega \backslash S \tag{42}$$

$$\mathcal{S}_l = \{i | f_i = \underline{\mathtt{f}}_i\}, \quad \mathcal{S}_l^c = \Omega \backslash \mathcal{S}_l \tag{43}$$

$$\mathcal{S}_u = \{i | f_i = \overline{\mathtt{f}}_i\}, \quad \mathcal{S}_u^c = \Omega \backslash \mathcal{S}_u \tag{44}$$

where $\Omega = \{i \in \mathbb{Z}^+ | i \le n\}$. Then the derivatives of f with respect to the inputs $\hat{\boldsymbol{f}}$, $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$ are as follows:

$$\frac{df_i}{d\hat{f}_j} = \begin{cases} 0, & i \in \mathcal{S}^c \text{ or } j \in \mathcal{S}^c \\ 1 - \frac{1}{n(\mathcal{S})}, & i = j \in \mathcal{S} \\ -\frac{1}{n(\mathcal{S})} & i \ne j, i \in \mathcal{S}, j \in \mathcal{S} \end{cases} \tag{45}$$

$$\frac{df_i}{d\underline{\mathtt{f}}_j} = \begin{cases} 0, & j \in \mathcal{S}_l, \forall i \in \Omega \\ 0, & j \in \mathcal{S}_l^c, i \in \mathcal{S}_l^c \backslash \{j\} \\ 1, & j \in \mathcal{S}_l^c, i = j \\ -\frac{1}{n(\mathcal{S}_l)} & j \in \mathcal{S}_l^c, i \in \mathcal{S}_l \end{cases} \qquad \frac{df_i}{d\overline{\mathtt{f}}_j} = \begin{cases} 0, & j \in \mathcal{S}_u, \forall i \in \Omega \\ 0, & j \in \mathcal{S}_u^c, i \in \mathcal{S}_u^c \backslash \{j\} \\ 1, & j \in \mathcal{S}_u^c, i = j \\ -\frac{1}{n(\mathcal{S}_u)} & j \in \mathcal{S}_u^c, i \in \mathcal{S}_u \end{cases} \tag{46}$$

B.5. Interval Bound Propagation through CBF-QP

Proposition B.1. *Consider a CBF-QP in the form of 40. Define function h_i to be $h_i : \boldsymbol{\eta}, \hat{\boldsymbol{f}} \mapsto f(\hat{\boldsymbol{f}})_i$. Given perturbed input in an interval bound $\underline{\boldsymbol{\eta}_i} \le \boldsymbol{\eta}_i \le \overline{\boldsymbol{\eta}_i}$, and $\underline{\hat{\boldsymbol{f}}_i} \le \hat{\boldsymbol{f}}_i \le \overline{\hat{\boldsymbol{f}}_i}$, we have*

$$h_i(\boldsymbol{\eta}_{ub}^i, \hat{\boldsymbol{f}}_{lb}^i) \le f(\hat{\boldsymbol{f}})_i \le h_i(\boldsymbol{\eta}_{lb}^i, \hat{\boldsymbol{f}}_{ub}^i) \tag{47}$$

where $\boldsymbol{\eta}^i_{ub}, \boldsymbol{\eta}^i_{lb}$ and $\hat{\boldsymbol{f}}^i_{ub}, \hat{\boldsymbol{f}}^i_{lb}$ are defined as follows:

$$\boldsymbol{\eta}^i_{ub} = \begin{cases} \overline{\boldsymbol{\eta}_j}, & j = i \\ \underline{\boldsymbol{\eta}_j}, & j \neq i \end{cases} \qquad \boldsymbol{\eta}^i_{lb} = \begin{cases} \underline{\boldsymbol{\eta}_j}, & j = i \\ \overline{\boldsymbol{\eta}_j}, & j \neq i \end{cases} \tag{48}$$

$$\hat{\boldsymbol{f}}^i_{ub} = \begin{cases} \overline{\hat{\boldsymbol{f}}_j}, & j = i \\ \underline{\hat{\boldsymbol{f}}_j}, & j \neq i \end{cases} \qquad \hat{\boldsymbol{f}}^i_{lb} = \begin{cases} \underline{\hat{\boldsymbol{f}}_j}, & j = i \\ \overline{\hat{\boldsymbol{f}}_j}, & j \neq i \end{cases} \tag{49}$$

Proof. We prove by contradiction. For the upper bound $f(\hat{\boldsymbol{f}})_i \leq h_i(\boldsymbol{\eta}^i_{lb}, \hat{\boldsymbol{f}}^i_{ub})$, suppose $f(\hat{\boldsymbol{f}})_i > h_i(\boldsymbol{\eta}^i_{lb}, \hat{\boldsymbol{f}}^i_{ub})$. Plug in Equation (41), we have

$$\left[\hat{\boldsymbol{f}}_i + \lambda\right]^{\overline{\mathtt{f}}(\boldsymbol{\eta}_i)}_{\underline{\mathtt{f}}(\boldsymbol{\eta}_i)} > \left[\overline{\hat{\boldsymbol{f}}_i} + \lambda'\right]^{\overline{\mathtt{f}}(\underline{\boldsymbol{\eta}_i})}_{\underline{\mathtt{f}}(\overline{\boldsymbol{\eta}_i})} \tag{50}$$

Since $\underline{\mathtt{f}}$ and $\overline{\mathtt{f}}$ are non-increasing function of $\boldsymbol{\eta}$, we have $\underline{\mathtt{f}}(\overline{\boldsymbol{\eta}_i}) \geq \underline{\mathtt{f}}(\boldsymbol{\eta}_i)$ and $\overline{\mathtt{f}}(\underline{\boldsymbol{\eta}_i}) \geq \overline{\mathtt{f}}(\boldsymbol{\eta}_i)$. Then $\hat{\boldsymbol{f}}_i + \lambda > \overline{\hat{\boldsymbol{f}}_i} + \lambda'$. Since $\hat{\boldsymbol{f}}_i \leq \overline{\hat{\boldsymbol{f}}_i}$, we have $\lambda > \lambda'$. Then for all $j \neq i$, we have

$$\left[\hat{\boldsymbol{f}}_j + \lambda\right]^{\overline{\mathtt{f}}(\boldsymbol{\eta}_j)}_{\underline{\mathtt{f}}(\boldsymbol{\eta}_j)} \geq \left[\underline{\hat{\boldsymbol{f}}_j} + \lambda'\right]^{\overline{\mathtt{f}}(\underline{\boldsymbol{\eta}_i})}_{\underline{\mathtt{f}}(\overline{\boldsymbol{\eta}_i})} \tag{51}$$

Sum on both sides of 50 and 51, we have

$$\mathbb{1}^\top f(\boldsymbol{\eta}, \hat{\boldsymbol{f}}) > \mathbb{1}^\top f(\boldsymbol{\eta}_{lb}, \hat{\boldsymbol{f}}_{ub}) \tag{52}$$

which is contradictory to the equality constraint in 40. Therefore, we have $f(\hat{\boldsymbol{f}})_i \leq h_i(\boldsymbol{\eta}^i_{lb}, \hat{\boldsymbol{f}}^i_{ub})$. The lower bound in 47 can be proved in the same way. □

C. Sampling Algorithms for Certification

C.1. Sampling on the decision boundary

We describe the process of generating samples on the decision boundary in Algorithm 2. The trick is to break down the n-class decision boundary sampling problem to 2 to $(n-1)$-class sampling problems. For instance, to generate samples with k non-zero elements on an n-class decision boundary with density T, one can sample points on an k-class decision boundary with density $T-k$ first, adding 1 to each dimension to make each element non-zero, and assign each element to an n dimensional vector. This operation is denoted by function G in Algorithm 2. G takes two list inputs a and c, increases each element in a by 1, rearranges the elements in a according to the indices given by c, and output a new list w of shape k. Equation 53 gives the form of the output of G. If the inputs are $y = 0, a = (3, 2, 3, 0), c = \{2, 3, 7\}$ and $k = 8$ (y is the label, a corresponds to a point on 4-class decision boundary, and c specifies the non-zero dimension except for the label dimension in an 8 dimensional vector), then the output is $w = (4, 0, 3, 4, 0, 0, 0, 1)$.

$$w_i = \begin{cases} a_0 + 1, & i = y \\ a_{|\{1,2,\cdots,i\}\cap c|} + 1, & i \in c \\ 0. & \text{o/w} \end{cases} \tag{53}$$

C.2. Rejection sampling

Mathematically, define the distance between two sets S_1, S_2 as $d(S_1, S_2) = \inf\{\|x-y\|_2 | x \in S_1, y \in S_2\}$. Let h be the barrier function, $L_0 = \{\boldsymbol{\eta} | h(\boldsymbol{\eta}) = 0\}$ be the 0-level set, and L_1, L_2 be the c_1-level set and c_2-level set that surrounds L_0, where $c_1 < 0 < c_2$. First, we sample mesh grids in a hypercube with fixed interval r along each dimension. Then we find c_1 and c_2 such that $d(L_1, L_0) \geq \frac{\sqrt{n}}{2} r$ and $d(L_2, L_0) \geq \frac{\sqrt{n}}{2} r$. Since the sampling interval is r, given a point $\boldsymbol{\eta} \in L_0$, there exists a point $\boldsymbol{\eta}^*$ from the mesh grids such that $|\boldsymbol{\eta}_i - \boldsymbol{\eta}^*_i| \leq \frac{r}{2}$ and $c_1 < h(\boldsymbol{\eta}^*) < c_2$. Therefore, we reject points with barrier function values less than c_1 or greater than c_2 and verify the Lyapunov condition holds around a hypercube with side length r for the mesh grids between L_1 and L_2. If the verification passes, the Lyapunov condition holds everywhere on L_0.

Algorithm 2 Sample the points on the decision boundary by dynamic programming.

```
Input: Number of classes $K$, sample density $T$, solution set $\mathrm{sol}$ with dimension $T \times K$.
// Initialize sol.
  Initialize each element of $\mathrm{sol}$ to be $\emptyset$.
  $\mathrm{sol}[0][k] = \{\mathbf{0}_k\}$, where $\mathbf{0}_k = [0, .., 0] \in \mathbb{R}^k$.
  $\mathrm{sol}[j][2] = \{[j/2, j/2]\}$.
  // Append elements to $\mathrm{sol}[j][k]$.
  for $j$ from 2 to $T$ do
      for $k$ from 3 to $K$ do
          for $l$ from 0 to $k-2$ do
              if $j - k + l \geq 0$ and $k - l \geq 0$ then
                  Let $\mathcal{C}$ be the set that contains $k-l-1$ combinations of $\{1, 2, ..., k-1\}$.
                  $\mathrm{sol}[j][k] = \mathrm{sol}[j][k] \cup \{G(y, a, c, k) | a \in \mathrm{sol}[j-k+l][k-l], c \in \mathcal{C}\}$.
return $\tilde{S}_y = \mathrm{sol}[T][K]/T$
```

D. Experiment Details

D.1. Image classification

To restrict the Lipschitz of the dynamics, we use orthogonal layers (Trockman & Kolter, 2021) in the neural network. Specifically, we have $\hat{\boldsymbol{f}}(\boldsymbol{\eta}, x) = W_3\sigma(W_2\sigma(W_1\boldsymbol{\eta} + g(x)) + b_2) + b_3$, where g is a neural network with 4 orthogonal convolution layers and 3 orthogonal linear layers, and W_1, W_2, W_3 are orthogonal matrices, σ is the ReLU activation function.

In the CBF-QP, we need to pick a class $\mathcal{K}_\infty^e$ function α for the inequality constraint $\mathtt{f} \geq -\alpha(\boldsymbol{\eta})$. Here we use $\alpha(\boldsymbol{\eta}) = c_1(e^{c_2\boldsymbol{\eta}} - 1)$, where $c_1 = 100$, and $c_2 = 0.02$. Comparing with a linear function, this $\alpha(\boldsymbol{\eta})$ leads to a higher margin over Lipschitz ratio, resulting in better certified accuracy.

During training, we train with batch size of 64. For each image, we sample 512 states. From epoch 1 to 10, all the states are uniformly sampled in the simplex. From epoch 11 to epoch 60, we linearly decay the proportion of uniform sampling in the simplex and increase the portion of uniform sampling within the correct classification set for each class. To sample uniformly in the simplex, we first sample n points from exponential distribution $\mathrm{Exp}(1)$ independently, then we normalize the n dimensional vector to have sum 1. To sample uniformly in the correct classification region for each class, we first uniformly sample from the simplex, then we swap the maximum element with the element corresponding to the correct label. We choose κ to be 2.0 in the loss function (eq. (10)). We use Adam optimizer with learning rate 0.01, and train for 300 epochs in total.

For certification, we choose $N = 40$ when sampling on the decision boundary. A larger N will lead to better certified accuracy but increases the computational cost dramatically. We ran the experiments on an NVIDIA RTX A6000 GPU.

D.2. Nonlinear control

The dynamics for the segway system is:

$$\frac{d}{dt}\begin{bmatrix} \phi \\ v \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \frac{\cos\phi(-1.8u+11.5v+9.8\sin\phi)-10.9u+68.4v-1.2\dot{\phi}^2\sin\phi}{\cos\phi-24.7} \\ \frac{(9.3u-58.8v)\cos\phi+38.6u-243.5v-\sin\phi(208.3+\dot{\phi}^2\cos\phi)}{\cos^2\phi-24.7} \end{bmatrix} \tag{54}$$

We use a 3 layer MLP as the controller. First, we train the controller to imitate a Linear Quadratic Regulator (LQR) controller. Then we jointly learn the Lyapunov function and finetune the weights in the MLP. Here we use standard linear layers rather than orthogonal layers, and control the empirical Lipschitz constant of the neural network by adversarial training.