Certifiably Robust Neural ODE with Learning-based Barrier Function

Runing Yang[1], Ruoxi Jia[1], Xiangyu Zhang[2], and Ming Jin[1,*]

***Abstract*— Neural Ordinary Differential Equations (ODEs) have gained traction in many applications. While recent studies have focused on empirically increasing the robustness of neural ODEs against natural or adversarial attacks, certified robustness is still lacking. In this work, we propose a framework for training a neural ODE using barrier functions and demonstrate improved robustness for classification problems. We further provide the *first generalization guarantee of robustness against adversarial attacks* using a wait-and-judge scenario approach.**

I. INTRODUCTION

Neural ordinary differential equations (NODEs) approximate nonlinear mappings by modeling the dynamics of hidden states by an ODE solver [7]. By extending discrete layerwise architectures to continuous-time dynamical systems, NODEs open up a wealth of applications, enjoy many desirable properties, such as parametric efficiency, constant memory requirements, and invertibility [7, 19, 23], and allow researchers to tap into the vast literature on control theory to achieve certain output requirements [19, 23].

This study aims to enhance the robustness of NODEs. Robustness is the ability of a model to maintain integrity in the face of input perturbations, either natural or adversarial. The vulnerability (lack of robustness) of deep learning to adversarial attacks is well-studied [18, 20, 8]. Recent study [21] indicates that, unlike conventional deep learning, NODEs have a certain level of inherent robustness due to the non-intersecting property of integral curves (see also [14] for another interesting view). Existing work to improve robustness can be classified (with overlaps) below:

1) *Regularization methods* inject random noise into each layer [17], randomly sample the end time of the ODE during training [17], or introduce additional penalty terms on weights [11];
2) *Control-theoretic approaches* design the training loss to promote certain properties of the dynamical system, such as steady-state constraint [21], contraction [23], reachability [13], and Lyapunov stability [16, 19].

While regularization methods are inspired by deep learning experiences (dropout, stochastic depth, and random smoothing in the case of [17, 11]), control-theoretic approaches directly promote certain aspects of the underlying dynamical systems—this is also the approach we take in this study.

Our key insight for classification problems, which are targeted in existing work, is that not all input-perturbation-induced output changes are adverse or need to be penalized; robustness depends only on those points *near the boundary* that are most *susceptible* to adversarial perturbations *along certain directions* (see Fig. 1 for an illustration, detailed in Sec. V-A). Hence, our method differs from the prior art in the following respects: instead of aiming to make the output for each data point to be a steady-state [21] or within a neighborhood of some Lyapunov-stable equilibrium [16, 19], we focus only on *vulnerable points* and penalize output changes along *adverse directions*. Our second insight lies in the difference between the notion of robustness for classification problems and similar concepts for control systems: for classification problems with a complex underlying mapping, outputs are *expected* to be different despite a small difference in inputs, which is not the case for robust/contractive systems [23]. To bring these insights to bear, we develop a framework that learns NODE parameters together with a robustness certificate based on control-barrier functions [1]. Our key contributions are as follows:

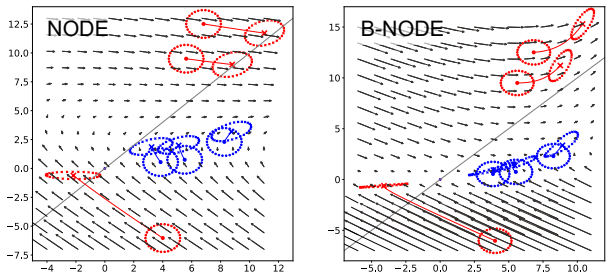

Fig. 1: Comparison between vanilla NODE and B-NODE (our method). The quiver plot shows the flows of trained NODEs, the propagation of the training data (solid lines), and how perturbed inputs (solid dots) are mapped to perturbed outputs (crosses) for both classes (red and blue). Grey line: decision boundary.

- Development of a framework for training robust NODEs using learning-based barrier functions;
- Establishment of the *first* generalization guarantee of robustness using a wait-and-judge approach;
- Demonstration of robustness against natural and adversarial perturbations for a range of benchmarks.

Contextualization of contributions. In comparison to studies that enhance the empirical robustness of NODEs [21, 17, 11, 23, 16, 19, 14], our study provides a rigorous analysis of the robustness guarantee. Unlike recent works on certified robustness that only analyze a *given* point [13, 15], we extend the certified robustness analysis to an *unseen in-distribution* point (Theorem 2). The difference can be interpreted as the extension from the training performance to the generalization

[1]Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA
[2]National Renewable Energy Laboratory, Golden, CO 80401, USA.
*Corresponding author: jinming@vt.edu

performance (i.e., from robust rate to robust probability in Def. 1). In our study, we pursue an approach based on scenario optimization [4, 5], while extending existing works in scenario optimization to the case of agnostic PAC (see a contemporary work that also achieves this [6]).

Next, we discuss preliminaries in Sec. II. Sec. III presents the robustness certificate based on barrier functions and the proposed robust training procedure. Theoretical analysis of robustness is performed in Sec. IV. Numerical results are discussed in Sec. V and the conclusion is drawn in Sec. VI. For the proof and additional experiments, please refer to [22].

Notations. We represent $[n] = \{1, ..., n\}$, $[k, n] = \{k, ..., n\}$, $[a]_+ = \max(0, a)$, and $\mathbb{1}(\cdot)$ as the indicator function (outputs 1 if the argument is true and 0 otherwise).

II. PRELIMINARIES

A. Neural ODEs

As a nonlinear mapping, a NODE specifies the relation between input $z(0) = \phi(x)$ and output $y = \psi(z(T), x)$ by the following differential equation:

$$\dot{z}(t) = f_\theta(z(t), t), \tag{1}$$

where x is the input data, $z(t)$ is the ODE state (hidden layer values) at time t, y is the output, and f_θ is a nonlinear function parameterized by θ.

The initial condition $z(0)$ is derived from the input by a feature mapping ϕ (i.e., input layer) and the output y is obtained from $z(T)$ through a function ψ (i.e., output layer). We slightly overload the notation $z(T, x)$ to make explicit the dependence of $z(T)$ on the initial state x. The input layer can be learned from scratch or fine-tuned based on some robust feature extractors [21, 16]. The output layer is a simple function, such as the softmax function for classification. Following common practice (e.g., [21]), we consider a time-invariant NODE (so we will write $f_\theta(z(t))$ throughout).

Classification. For classification problems, an affine transformation is applied to the final state of NODE to obtain an embedding $Wz(T) \in \mathbb{R}^K$ of the same dimension as the number of classes K, where $W \in \mathbb{R}^{K \times d}$ is the parameters to be learned. Then, the embedding can be passed through either an argmax function $\arg\max_k [Wz(T)]_k$, where $[Wz(T)]_k$ is the k-th coordinate of the vector, or a softmax function to obtain a probability vector. To make explicit the dependence on the input data and the neural ODE system, we represent the output $y = \psi(z(T), x) = g_W(x, f_\theta)$. Unless otherwise specified, we denote θ to also include the parameter W and omit the dependency of g_W on W for notational simplicity.

Forward/backward process. Forward pass can be performed using standard ODE solvers (such as Euler's method or Runge–Kutta method) to obtain the final state $z(T)$. For training, we need to calculate the derivative of the loss function with respect to NODE parameters, which can be obtained by either auto-differentiation or the adjoint sensitivity method [7].

B. Attack model and adversarial training

Attack model. In this study, we consider evasion attacks, where the attacks occur at the time of inference after the model has already been trained [12].

We consider a range of capabilities, from a random attack scenario, where noise of certain variance is added to the input data, to a whitebox attack, where the attacker can have full access to model parameters to reduce model performance.

Adversarial training. Suppose we have a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i \in [n]}$. Standard training solves the following empirical risk minimization (ERM) problem:

$$\min_\theta \sum_{(x,y) \in \mathcal{D}} \ell(g(x, f_\theta), y), \tag{2}$$

where ℓ is the loss function, such as the cross-entropy loss. A family of adversarial training techniques can be seen as solving the following min-max optimization:

$$\min_\theta \left\{ \sum_{(x,y) \in \mathcal{D}} \max_{\delta \in \Delta} \ell(g(x + \delta, f_\theta), y) \right\}, \tag{3}$$

where δ is the perturbation, Δ is the set of feasible perturbations (e.g., vectors with bounded ℓ_1, ℓ_2, or ℓ_∞ norm). Typical algorithms to solve inner maximization include projected gradient descent (PGD) [18] and fast sign gradient method (FSGM) [12]. During each iteration of the outer minimization, the inner maximization is approximated by either PGD or FSGM. Let $\mathcal{D}_\theta = \{(x'_i, y_i)\}_{i \in [n]}$ denote the adversarial data, where x'_i is obtained by some attack algorithm (e.g., PGD, FSGM) on the data (x, y). Then, the adversarial training procedure can be seen as performing standard training on augmented datasets:

$$\min_\theta \sum_{(x,y) \in \mathcal{D} \cup \mathcal{D}_\theta} \ell(g(x, f_\theta), y), \tag{4}$$

where $\mathcal{D}_\theta$ is a set that depends on both the current model θ and the attack algorithm. In our experiments, we also consider generating $\mathcal{D}_\theta$ by simply adding Gaussian noises to each input to emulate natural perturbations, in which case the dependence on θ is no longer needed.

III. METHODOLOGY

A. Robustness certificate

Robust set. Suppose that the input x takes values in a compact metric space $\mathcal{X}$ and the output $y \in \mathcal{Y} = \{1, ..., K\}$. For a given x and a fixed NODE with parameter θ, let $\Xi_\theta(x) = \{z(T) : z(0) = x + \delta, \dot{z}(t) = f_\theta(z(t)), t \in [0, T], \delta \in \Delta\}$ denote the *output perturbation set*, i.e., the set of final states of NODE when the initial state is perturbed by any $\delta \in \Delta$. Let $\mathcal{C}_y = \{z : [Wz]_y \geq [Wz]_{y'}, \forall y' \in \mathcal{Y}\}$ represent the set of embeddings that lead to the selection of class y under the argmax rule applied after an affine transformation. Then, for any data sample (x, y), the *robust set* $\mathcal{S}_\theta(x, y)$ is given by

$$\mathcal{S}_\theta(x, y) \coloneqq \Xi_\theta(x) \cap \mathcal{C}_y, \tag{5}$$

which represents the set of perturbed final states that still lead to a correct result. Consider the following two cases:

1) $\Xi_\theta(x) \setminus \mathcal{S}_\theta(x,y) \neq \emptyset$ implies that there is a $\delta \in \Delta$ that falsifies the output, i.e., $\arg\max_k [Wz(T)]_k \neq y$;
2) $\Xi_\theta(x) = \mathcal{S}_\theta(x,y)$ implies that NODE θ can provide a correct decision under any perturbation.

We also note that $\mathcal{S}_\theta(x,y) \subseteq \mathcal{C}_y$ always holds.

Robustness certificate. Recall that an extended class $\mathcal{K}_\infty$ function is a function $\alpha : \mathbb{R} \to \mathbb{R}$ that is strictly increasing and with $\alpha(0) = 0$. The following result provides a certificate for a given point (x, y) against arbitrary perturbation $\delta \in \Delta$.

Theorem 1 (Robustness certificate)**.** *Suppose there exists a continuous and almost everywhere differentiable function $h_y : \mathcal{C}_y \to \mathbb{R}$ such that **(1)** $\mathcal{S}_\theta(x,y) \subseteq \mathcal{H}_y \subseteq \mathcal{C}_y$, where $\mathcal{H}_y = \{z : h_y(z) \geq 0\}$, and **(2)** there exists an extended class $\mathcal{K}_\infty$ function α such that for NODE* (1)*:*

$$\dot{h}_y(z(t)) \geq -\alpha(h_y(z(t))) \tag{6}$$

for all $z(t) \in \Xi_\theta(x)$ and $t \in [0, T]$, then, for any $\delta \in \Delta$, there exists a finite $T' > 0$ such that $z(t, x+\delta) \in \mathcal{C}_y$ for all $t \geq T'$. We call the function h_y a robustness certificate (for point (x, y) under NODE θ).

Remark 1. A natural candidate for h_y is the family of functions parameterized by W: $h_y(z) = [Wz]_y - \max_{k \neq y} [Wz]_k$, which is continuous, differentiable everywhere, and satisfies condition (1) because in this case $\mathcal{H}_y = \mathcal{C}_y$.

Remark 2. The robustness certificate defined above enforces the invariance of $\mathcal{H}_y$ that includes the robust set $\mathcal{S}_\theta(x,y)$ but excludes the remaining perturbation set $\Xi_\theta(x) \setminus \mathcal{S}_\theta(x,y)$ that corresponds to a corrupted result. In addition, it also ensures that for perturbed final states in $\Xi_\theta(x) \setminus \mathcal{S}_\theta(x,y)$, the NODE will be able to recover to a correct decision asymptotically, that is, $S_\theta(x,y) \subseteq \mathcal{C}_y$ if we run the underlying NODE long enough (with T large enough).

The following corollary extends the certificate to a dataset.

Corollary 1. *Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i \in [n]}$, suppose there exists a continuous and almost everywhere differentiable function $h_y : \mathcal{C}_y \to \mathbb{R}$ such that satisfies conditions (1) and (2) in Theorem 1. Then, for any $(x, y) \in \mathcal{D}$ and any $\delta \in \Delta$, there exists a finite $T' > 0$ such that $z(t, x+\delta) \in \mathcal{C}_y$ for all $t \geq T'$.*

Remark 3. While barrier functions have been widely used in control systems to establish safety, avoidance, or eventuality properties [1], this study is the first to adapt the method for a classification problem with new notions of robust set and robustness certificate.

B. Training with robustness certificate

To integrate the robustness certificate into training, ideally, we can solve the following optimization:

$$\min_{\theta, \{h_y\}_{y \in \mathcal{Y}}} \sum_{(x,y) \in \mathcal{D}} \ell(g(x, f_\theta), y), \tag{7a}$$

$$\text{s.t.} \quad h_y \in C^1(\mathbb{R}^d, \mathbb{R}), \quad \forall y \in \mathcal{Y} \tag{7b}$$

$$\{z : h_y(z) \geq 0\} \subseteq \mathcal{C}_y, \quad \forall y \in \mathcal{Y} \tag{7c}$$

$$\mathcal{S}_\theta(x,y) \subseteq \{z : h_y(z) \geq 0\}, \quad \forall (x,y) \in \mathcal{D} \tag{7d}$$

$$\dot{h}_y(z(t)) \geq -\alpha(h_y(z(t))), \quad \forall z(t) \in \Xi_\theta(x), (x,y) \in \mathcal{D} \tag{7e}$$

where objective (7a) can be the usual cross-entropy loss, (7b) enforces that h_y is continuous and almost everywhere differentiable, (7c), (7d) and (7e) correspond to conditions (1) and (2) in Corollary 1, respectively. For comparison, TisODE [21] and SODEF [16] impose stability or steady-state constraints on NODE dynamics f_θ, whereas robustness is enforced by the existence of a certificate function.

Learning-based certificate. Since optimization over functions $\{h_y\}_{y \in \mathcal{Y}}$ can be intractable, we specify a parametric form of $h_y(z) = [Wz]_y - \max_{k \neq y} [Wz]_k$ for a given W shared among $\{h_y\}_{y \in \mathcal{Y}}$. This immediately satisfies constraints (7b)–(7d) (see Remark 1). We also tie this parameter to the parameter of g_W because their goals are aligned; in this case, the certificate coincides with the last layer transformation. Note that constraint (7e) is specified over a set. While sum-of-squares provides a principled approach [1], computation becomes quickly intractable for higher-dimensional systems. Along the line of learning-based certificates [9], we propose two complementary ways to approximate this constraint with data:

- Random samples: For each data point (x, y), we randomly sample a set of perturbations; for each perturbation δ, we forward propagate NODE to obtain $z(T)$. The set of such $z(T)$'s is collected by $\hat{\Xi}_\theta(x)$, which is used to replace $\Xi_\theta(x)$ in (7e).
- Adversarial examples: Similar to the above, except that for each data (x, y), we apply an attack algorithm to find a set of δ's. The corresponding $z(T)$'s together with the original $z(T)$ without input perturbation are collected by $\hat{\Xi}_\theta(x)$ used to replace $\Xi_\theta(x)$ in (7e).

In general, we can expect that a sufficient number of random samples can bring $\hat{\Xi}_\theta(x)$ close to $\Xi_\theta(x)$; in the case of adversarial samples, the constraint is more biased towards attack cases, where the robustness certificate is most likely to be violated. We use $\mathcal{D}_\theta(x,y)$ and $\mathcal{D}_\theta = \cup_{(x,y) \in \mathcal{D}} \mathcal{D}_\theta(x,y)$ to represent perturbed input sets for each data (x, y) and the entire dataset, respectively, which can be a mixture of random samples and adversarial examples. Therefore, instead of directly optimizing (7), in our implementation, we optimize the following empirical Lagrangian:

$$\min_\theta \quad \mathcal{L}(\theta) \coloneqq \mathcal{L}_0(\theta) + \lambda_1 \mathcal{L}_1(\theta) + \lambda_2 \mathcal{L}_2(\theta) + \lambda_3 \mathcal{L}_3(\theta), \tag{8}$$

where $\mathcal{L}_0(\theta) = \sum_{(x,y) \in \mathcal{D}} \ell(g(x, f_\theta), y)$ is the usual training

loss, $\mathcal{L}_1(\theta) = \sum_{(x,y)\in\mathcal{D}_\theta}[-h_y(z(T,x))+\epsilon_1]_+$ is the loss that penalizes mistakes due to random/adversarial perturbations, $\mathcal{L}_2(\theta) = \sum_{(x,y)\in\mathcal{D}\cup\mathcal{D}_\theta}[-\dot{h}_y(z(T,x)) - h_y(z(T,x)) + \epsilon_2]_+$ is the regularization term for constraint (7e) with the simple choice of $\alpha(a) = a$, $\mathcal{L}_3(\theta) = \sum_{(x,y)\in\mathcal{D}\cup\mathcal{D}_\theta}[|\dot{h}_y(z(T,x))| - \epsilon_3]_+$ for some constant $\{\epsilon_i > 0\}_{i\in[3]}$, and $\{\lambda_i \geq 0\}_{i\in[3]}$ are regularization coefficients (can be seen as dual variables for the Lagrangian relaxation of (7)).

Remark 4 (Reverse-time robustness.). The term $\mathcal{L}_3(\theta)$ is not needed if we are able to enforce constraint (7e) exactly (i.e., over the entire set $\Xi_\theta(x)$). However, it is necessary in the case of sample-based constraints. Intuitively, consider an input (x, y) that leads to a final state $z(T, x)$ lies within but close to the boundary of $\{z : h_y(z) \geq 0\}$. If the rate $|\dot{h}(z(T,x))|$ is large and if the constraint (7e) is not enforced on all points along the trajectory $z(t) \in \Xi_\theta(x)$, then it is plausible that a small time shift $t' = t-\delta_t$ can drive the state out of $\mathcal{S}_\theta(x, y)$. Such an attack, called revserse-time attack, has not been discussed in the literature due to the specific nature of NODE; nevertheless, it can be performed easily because perturbation on the input $\delta = z(0, x) - z(-\delta_t, x)$ can be small if δ_t is small (here, $z(-\delta_t, x)$ is the state after running NODE for δ_t in reverse time).

IV. THEORETICAL ANALYSIS

We present a theoretical framework to analyze to what extent the level of robustness of a trained NODE generalizes to unseen samples from the same distribution.

A. Certified robustness bound

Let $\xi_i \coloneqq (x_i, y_i)$ be a random sample drawn from probability space $(\Omega, \mathcal{F}, \mathbb{P})$, which is endowed with a σ-algebra $\mathcal{F}$ and a probability measure $\mathbb{P}$. A dataset $\mathcal{D}_n \coloneqq \{\xi_i\}_{i\in[n]} \in \Omega^n$ consists of n observations drawn independently from Ω according to $\mathbb{P}$. Formally, solving (8) for a given dataset can be regarded as a numerical procedure $\mathcal{A}_r : \Omega^r \to \Theta$, indexed by the sample size $r \in [n]$, which returns a NODE parameter θ along with a robustness certificate. We use $\mathcal{A}(S)$ to denote $\mathcal{A}_r(S)$ for any subset $S \subseteq \mathcal{D}_n$, where we omit the subscript $r = |S|$, i.e., the cardinality of S. For every data sample $\xi = (x, y)$, define

$$\Theta_\xi \coloneqq \Big\{\theta \in\Theta : h_y(z(T,x)) \geq 0, |\dot{h}_y(z(t))| \leq \epsilon, \\ \dot{h}_y(z(t)) \geq -\alpha(h_y(z(t))), \forall z(t) \in \Xi_\theta(x)\Big\}$$

as the set of NODE parameters that renders the data ξ correctly classified (due to $h_y(z(T, x)) \geq 0$) and *certifiably robust* to any perturbation in Δ (due to Theorem 1). Similarly, we define an empirical estimate of the set $\hat{\Theta}_\xi$ by replacing the conditions above with $h_y(z(T,x)) \geq \epsilon_1, \dot{h}_y(z(t)) \geq -\alpha(h_y(z(t)))+\epsilon_2$, and $|\dot{h}_y(z(t))| \leq \epsilon_3$ for all $z(t) \in \hat{\Xi}_\theta(x)$, which coincide with the loss terms in $\{\mathcal{L}_i(\theta)\}_{i\in[3]}$ for (x, y).

The goal of the analysis is to study how the robustness of a NODE $\theta_n = \mathcal{A}(\mathcal{D}_n)$ returned by $\mathcal{A}$ on a dataset $\mathcal{D}_n$ generalizes to a yet unseen data $\xi \in \Omega$.

Definition 1. The robustness probability (RP) of a given parameter $\theta \in \Theta$ is defined as $V(\theta) \coloneqq \mathbb{P}(\xi \in \Omega : \theta \in \Theta_\xi)$. The robustness rate (RR) on an evaluation subset $S \subseteq \mathcal{D}_n$ is $\hat{V}(\theta; S) \coloneqq \frac{1}{n}\sum_{\xi\in\mathcal{D}_n} \mathbb{1}(\theta \in \hat{\Theta}_\xi) - \frac{1}{n}\sum_{\xi\in\mathcal{D}_n\setminus S} \mathbb{1}(\theta \notin \hat{\Theta}_\xi)$.

Note that RP $V(\theta)$ should be interpreted as a *lower bound* on the true robustness probability of a given NODE θ, since $\theta \in \Theta_\xi$ implies that the data ξ is certifiably robust to all $\delta \in \Delta$ but a data ξ' may be robust even if $\theta \notin \Theta_{\xi'}$ (i.e., the condition $\theta \in \Theta_\xi$ is a sufficient but not necessary condition for certified robustness). We use $\hat{V}(\delta)$ for $\hat{V}(\delta; \mathcal{D}_n)$ if the evaluation dataset is $\mathcal{D}_n$ for notational simplicity.

We make the following assumptions.

Assumption 1. *For any data $\xi \in \mathcal{D}_n$ in the training set, we have that $\hat{\Theta}_\xi \subseteq \Theta_\xi$.*

Assumption 2. *The algorithm $\mathcal{A} : \Omega^n \to \Theta$ to solve* (8) *yields a unique output. Suppose the output is $\theta_n = \mathcal{A}(\mathcal{D}_n)$. For any subset $S \subseteq \mathcal{D}_n$ where the membership $\xi \in S$ implies that $\theta_n \in \hat{\Theta}_\xi$, we have that $\theta_n = \mathcal{A}(\mathcal{D}_n \setminus S)$.*

Assumption 1 implies that a point that is shown to be robust against adversarial examples is robust for all perturbations in Δ; under the condition that NODE dynamics f_θ is bounded, this can be satisfied by choosing margins $\{\epsilon_i\}_{[i\in[3]}$ large enough. The uniqueness requirement in Assumption 2 can be satisfied by a simple tie-break rule in the case of multiple solutions, e.g., selecting the one with the minimum norm. The second requirement is reminiscent of support vector machines: if we consider the set S defined there as non-support data, the assumption stipulates that removing the non-support data does not change the solution.[1]

Main result. In the following, we focus the analysis on proving a PAC (probably approximately correct)-type of result: $V(\theta_n) \geq \hat{V}(\theta_n) - \kappa$ for some parameter $\kappa \in (0, 1)$ with a probability at least $1 - \beta$. A probability bound of $1 - \beta$ is necessary as $\theta_n = \mathcal{A}(\mathcal{D}_n)$ is a random variable defined over Ω^n. Before stating our main result, consider a function $\mathcal{I}_n : \Omega^n \to \{1, .., n\}$ that returns $\mathcal{S} = \mathcal{I}_n(\mathcal{D}_n)$ for a given dataset $\mathcal{D}_n$, where $\mathcal{S} \subseteq \{1, ..., n\}$ is a subset of indices such that $\mathcal{D}_n(\mathcal{S})$, namely, the subset of data indexed by $\mathcal{S}$, has the property that $\mathcal{A}(\mathcal{D}_n) = \mathcal{A}(\mathcal{D}_n(\mathcal{S}))$ and $\hat{V}(\mathcal{A}(\mathcal{D}_n); \mathcal{D}_n) = \hat{V}(\mathcal{A}(\mathcal{D}_n(\mathcal{S})); \mathcal{D}_n(\mathcal{S}))$.[2]

Theorem 2. *For the given $\mathcal{A}$ that solves* (8)*, it holds that*

$$\mathbb{P}^n\Big(V(\theta_n) \geq \hat{V}(\theta_n) - \kappa(r_n)\Big) \geq 1 - \beta, \tag{9}$$

where $r_n = |\mathcal{I}_n(\mathcal{D}_n)|$ and $\theta_n = \mathcal{A}(\mathcal{D}_n)$. Here, $\kappa(r) : \{0, \ldots, n\} \to [0, 1]$ is any function such that $\kappa(0) = 1$ and

$$\sum_{r=1}^{n} \binom{n}{r} (1 - \kappa(r))^{n-r} \leq \beta.$$

[1]Indeed, for these data in S, given the obtained solution θ_n, the terms in $\{\mathcal{L}_i(\theta_n)\}_{i\in[3]}$ corresponding to these data are zero.

[2]As an example of such function, we can construct a set S such that $\xi_i \in S$ if and only if $\xi_i \in \mathcal{D}_n$ and $\theta_n \notin \Theta_{\xi_i}$. We collect the original indices of all points in S as $\mathcal{S}$ as the output of $\mathcal{I}_n(\mathcal{D}_n)$. The validity of this function is implied by Assumption 2. However, note that the existence of such a function does not rely on Assumption 2.

Remark 5. To compute the bound, a simple choice is to split β evenly among the n terms in the summation:

$$\kappa(r) = \begin{cases} 1 & \text{if } r = n \\ \left[1 - \left(\frac{\beta}{n\binom{n}{r}}\right)^{1/(n-r)}\right]_+ & \text{o.w.} \end{cases}, \quad (10)$$

where $\binom{n}{r}$ is the n-choose-r binomial coefficient.

Remark 6. Note that (8) is a computationally tractable *relaxation* of (7) from constrained optimization with expectation constraints to unconstrained optimization with empirical data. Theorem 2 is derived for the solution obtained by (8), where the robustness is evaluated against the constraints from (7). In particular, for a given NODE parameter θ, a point ξ that satisfies the constraints from (7) (i.e., $\theta \in \Theta_\xi$) is certifiably robust by Theorem 1. Hence, a point ξ that satisfies the empirical version $\theta \in \hat{\Theta}_\xi$ (which leads to zero loss incorporated in (8)) is certifiably robust (in view of Assumption 1). Since Theorem 2 establishes a lower bound of the RP of the solution obtained from (8) using RR, this lower bound also holds according to the constraints from (7).

Remark 7. The theoretical bound in (9) differs from common generalization bounds in learning theory in that the bound is evaluated a-posteriori after the solution is computed. This "wait-and-judge" type of result has been developed in convex optimization [4]. In addition, we note that existing methods apply primarily to feasibility problems [4, 5], which translate to the assumption that there exists a NODE parameter $\theta \in \Theta$ such that $\theta \in \Theta_\xi$ for all $\xi \in \mathcal{D}_n$. In other words, they require the assumption to achieve zero loss during training of (8). This is unrealistic because we *cannot* expect a NODE that is robust for *all* data. Our method is broadly applicable to the relaxed setting. To put into context, the difference is analogous to the extension of standard PAC learning to the agnostic PAC theory, or the extension of the example-consistent framework to the non-consistent schemes [6].

V. EXPERIMENTS

In this section, we demonstrate the effectiveness of our framework for both non-adversarial and adversarial robustness. For additional experimental details, including ODE architecture and attack specifications, as well as further results, please refer to the online version [22].

A. 2D Binary classification

For each data shown in Fig. 1, 30 perturbed inputs are uniformly sampled within a radius of 1.2 centered on the original data point (red and blue dots along the circles).

Results. In Fig. 1, both methods correctly classify all original points, but vanilla NODE's output regions intersect the decision boundary, indicating potential misclassifications under perturbation, which is not observed for B-NODE. B-NODE generally has smaller perturbed output regions, with their main axis almost parallel to the decision boundary, indicating successful transformation of anisotropic perturbations on inputs into isotropic perturbations on outputs, where most perturbations are innocuous (i.e., do not change decisions). In contrast, vanilla NODE is more susceptible to output perturbations across decision boundaries (in some cases, even running the integral longer can lead to wrong decisions).

B. Evaluating robustness on MNIST

First, we test the improvement of robustness against Gaussian noise (Table I). While vanilla NODE is robust under mild perturbations, the performance drops significantly as the variance of Gaussian noise increases.

TABLE I: Robustness against Gaussian noise (zero mean and different variances σ^2). The average and standard deviation are across 10 random perturbations for each data.

	$\sigma^2 = 20$	$\sigma^2 = 50$	$\sigma^2 = 100$	$\sigma^2 = 150$
vanilla NODE	99.8±0.1	99.7±0.1	82.7±0.5	53.7±0.5
B-NODE	99.8±0.1	99.7±0.1	98.4±0.2	89.4±0.6

Robustness against PGD attack [18] is shown in TablevII. We compare our method to the vanilla NODE trained with clean data [7], NODE trained with data augmentation (AT-NODE), and ODE-TRADES [24], using the same architecture for both methods to ensure a fair comparison. As the attack radius increases (ℓ_∞-norm bounds shown in the first row of each table), the performance deteriorates for all methods. However, the drop in performance for B-NODE is much less, indicating enhanced robustness.

TABLE II: Robustness against PGD attack on MNIST.

PGD	0.01	0.02	0.03	0.04	0.05
vanilla NODE	86.14±1.33	79.26±1.80	71.04±2.01	60.36±2.34	47.40±2.11
AT-NODE	91.76±1.16	89.34±1.14	85.8±1.94	82.94±1.64	78.38±1.67
ODE-TRADES	91.46±1.07	88.98±1.98	85.08±1.61	81.94±0.94	77.32±1.89
B-NODE	**92.62±0.87**	**91.52±1.22**	**89.16±1.25**	**86.36±1.78**	**83.18±1.75**

According to [14], evaluating against PGD is not sufficient to empirically demonstrate the adversarial robustness of NODEs because NODEs can obfuscate gradients. Therefore, additional experiments against AutoPGD [8] and Square Attack [2] are performed to demonstrate effectiveness of defense, as shown in Tables III and IV.

TABLE III: Robustness against AutoPGD attack on MNIST.

AutoPGD	0.01	0.02	0.03	0.04	0.05
vanilla NODE	84.41±1.11	76.66±1.65	68.99±1.71	58.33±2.16	46.31±2.51
AT-NODE	91.34±1.51	89.49±1.89	87.60±2.67	82.70±2.00	78.35±1.87
ODE-TRADES	92.19±1.92	88.30±2.30	85.31±2.41	81.90±1.85	76.89±3.85
B-NODE	**93.45±1.42**	**91.25±1.34**	**88.95±2.09**	**86.30±3.24**	**82.65±2.58**

TABLE IV: Robustness against Square attack on MNIST.

Square attack	0.01	0.02	0.03	0.04	0.05
vanilla NODE	86.55±2.65	83.29±3.66	72.50±5.85	62.49±3.59	51.58±4.05
AT-NODE	92.65±1.72	90.99±2.50	89.50±3.84	85.51±3.84	81.13±5.44
ODE-TRADES	92.06±3.10	91.85±2.73	90.08±2.73	86.24±2.56	82.41±2.96
B-NODE	**94.49±2.55**	**92.99±0.99**	**91.52±3.49**	**88.51±4.51**	**85.49±2.50**

C. Evaluation on FashionMNIST and CIFAR10

For the FashionMNIST data, we tested with convolution Neural ODEs, which uses a 2D matrix to represent hidden states during forward propagation. AT-NODE, ODE-TRADES and B-NODE are trained with clean data augmented with adversarial examples, which are generated by 40 steps L_∞ norm PGD attack, whose magnitude ε is 8/255. The clean data accuracy for vanilla NODE, AT-NODE, B-NODE and ODE-TRADES are 85.36%, 82.10%, 82.68% and 83.24%, respectively.

TABLE V: Performance on FashionMNIST.

	PGD-$\frac{4}{255}$	PGD-$\frac{6}{255}$	PGD-$\frac{8}{255}$	PGD-$\frac{10}{255}$	PGD-$\frac{12}{255}$
AT-NODE	75.99±2.28	72.15±2.87	70.70±2.81	66.25±2.67	63.85±3.21
ODE-TRADES	77.68±2.25	75.47±1.70	72.53±1.94	68.32±1.94	67.12±2.21
B-NODE	**78.96±2.48**	**76.32±2.67**	**73.62±2.21**	**69.12±2.69**	**67.36±1.99**

For the CIFAR10 experiment, we use a pre-trained CNN model for feature extractor, the output of which is provided to NODE as an initial state [16]. The clean data accuracy for vanilla NODE, B-NODE, ODE-TRADES are 89.68%, 89.16%, and 90.48%, respectively.

TABLE VI: Performance on CIFAR10.

	PGD-$\frac{6}{255}$	PGD-$\frac{9}{255}$	PGD-$\frac{12}{255}$	PGD-$\frac{15}{255}$
AT-NODE	58.56±1.65	45.35±2.46	35.71±2.41	25.76±3.12
ODE-TRADES	59.28±2.36	47.60±2.16	39.48±2.93	30.48±2.84
B-NODE	**61.81±2.35**	**48.52±2.45**	**39.60±3.14**	**30.52±2.27**

The results in Tables V and VI indicate that B-NODE exhibits the best performance among all methods. Nevertheless, we observed a significant decrease in performance compared to the accuracy on clean data, suggesting that there is room for improvement in future work.

D. Ablation analysis

We conducted an experiment to analyze the impact of learning-based barrier functions and data augmentation on the robustness of four different models (see details in [22, Appendix]). As shown in Table IX, barrier function without data augmentation can already improve robustness, sometimes by 15% (in the case of PGD-0.05). It is remarkable to see improvement in all cases of attacks, indicating some level of "universal robustness." However, the use of the barrier function in conjunction with data augmentation yields the most robust performance.

TABLE VII: Ablation analysis for models with and without barrier function loss or data augmentation on MNIST dataset.

	PGD-0.01	PGD-0.02	PGD-0.03	PGD-0.04	PGD-0.05
vanilla NODE	86.14±1.33	79.26±1.80	71.04±2.01	60.36±2.34	47.40±2.11
B-NODE w/o AT	88.62±1.40	83.82±1.10	79.00±1.39	71.22±2.23	61.46±1.64
AT-NODE	91.76±1.16	89.34±1.14	85.80±1.94	82.94±1.64	78.38±1.67
B-NODE w/ AT	92.62±0.87	91.52±1.217	89.16±1.25	86.36±1.78	83.18±1.75

VI. CONCLUSIONS

We have developed an algorithm to train NODEs based on barrier functions with certified robustness. Future directions include examining the necessity of the robustness certificates and exploring the incorporation of (exponential) control barrier functions to problems beyond classifications.

REFERENCES

[1] A. Ames et al. "Control barrier functions: Theory and applications". In: *2019 18th European control conference (ECC)*. IEEE. 2019, pp. 3420–3431.

[2] Maksym Andriushchenko et al. "Square attack: a query-efficient black-box adversarial attack via random search". In: *ECCV*. Springer. 2020, pp. 484–501.

[3] F. Blanchini. "Set invariance in control". In: *Automatica* 35.11 (1999), pp. 1747–1767.

[4] M. Campi and S. Garatti. "Wait-and-judge scenario optimization". In: *Mathematical Programming* 167.1 (2018), pp. 155–189.

[5] M. Campi, S. Garatti, and F. Ramponi. "A general scenario theory for nonconvex optimization and decision making". In: *IEEE Transactions on Automatic Control* 63.12 (2018), pp. 4067–4078.

[6] Marco C Campi and Simone Garatti. "Compression, Generalization and Learning". In: *arXiv preprint arXiv:2301.12767* (2023).

[7] R. Chen et al. "Neural ordinary differential equations". In: *NeurIPS* 31 (2018).

[8] Francesco Croce and Matthias Hein. "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *ICML*. PMLR. 2020, pp. 2206–2216.

[9] C. Dawson, S. Gao, and C. Fan. "Safe Control with Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction methods". In: *arXiv preprint arXiv:2202.11762* (2022).

[10] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. "Augmented neural odes". In: *Advances in Neural Information Processing Systems* 32 (2019).

[11] A. Ghosh et al. "Steer: Simple temporal regularization for neural ode". In: *NeurIPS* 33 (2020), pp. 14831–14843.

[12] I. Goodfellow, J. Shlens, and C. Szegedy. "Explaining and harnessing adversarial examples". In: *ICLR* (2015).

[13] Sophie Grunbacher et al. "On the verification of neural odes with stochastic guarantees". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 13. 2021, pp. 11525–11535.

[14] Yifei Huang et al. "Adversarial Robustness of Stabilized Neural ODE Might be from Obfuscated Gradients". In: *Mathematical and Scientific Machine Learning*. PMLR. 2022, pp. 497–515.

[15] Yujia Huang et al. "FI-ODE: Certified and Robust Forward Invariance in Neural ODEs". In: *arXiv preprint arXiv:2210.16940* (2022).

[16] Q. Kang et al. "Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks". In: *NeurIPS* 34 (2021), pp. 14925–14937.

[17] X. Liu et al. "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*. 2020, pp. 282–290.

[18] A. Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *ICLR* (2018).

[19] I. Rodriguez, A. Ames, and Y. Yue. "LyaNet: A Lyapunov framework for training neural ODEs". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 18687–18703.

[20] Vincent Tjeng, Kai Y Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *International Conference on Learning Representations*. 2018.

[21] H. Yan et al. "On robustness of neural ordinary differential equations". In: *ICLR* (2020).

[22] R. Yang et al. *Certifiably Robust Neural ODE with Learning-based Barrier Function*. Link: `http://www.jinming.tech/papers/B-NODE22.pdf`. 2022.

[23] M. Zakwan, L. Xu, and G. Ferrari-Trecate. "On Robust Classification using Contractive Hamiltonian Neural ODEs". In: *arXiv preprint arXiv:2203.11805* (2022).

[24] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *International conference on machine learning*. PMLR. 2019, pp. 7472–7482.

APPENDIX

A. Proof of Theorem 1

Since for any $z \in \partial\mathcal{H}_y$ (boundary of $\mathcal{H}_y$), $\dot{h}_y(z) \geq -\alpha(h_y(z)) = 0$, the set $\mathcal{H}_y$ is forward invariant (Nagumo's theorem [3]). Let $\bar{\Delta} = \{\delta \in \Delta : z(0) = x + \delta, \dot{z}(t) = f_\theta(z(t)), t \in [0, T], z(T) \in \mathcal{S}_\theta(x, y)\}$ denote the set of *innocuous perturbations* that do not lead to a false prediction. Then, for any $\delta \in \bar{\Delta}$, we have that $z(t, x + \delta) \in \mathcal{H}_y \subseteq \mathcal{C}_y$ for any $t \geq T$. For the remaining $\delta \in \Delta \setminus \bar{\Delta}$, we know that $z(T, x + \delta) \in \Xi_\theta(x) \setminus \mathcal{S}_\theta(x, y)$ by definition. Condition (1) implies that $(\Xi_\theta(x) \setminus \mathcal{S}_\theta(x, y)) \cap \mathcal{H}_y = \emptyset$. By condition (2), the set $\mathcal{H}_y$ is asymptotically stable. Hence, there exists an $T' \geq T$ such that $z(t, x + \delta) \in \mathcal{C}_y$ for any $\delta \in \Delta \setminus \bar{\Delta}$. This concludes the proof.

B. Proof of Theorem 2

We write $\mathcal{A}(\mathcal{D}_n)$ for δ_n to make explicit the dependence on a random dataset $\mathcal{D}_n$. To proceed, define

$$\Lambda = \left\{\mathcal{D}_n \in \Omega^n : V(\mathcal{A}(\mathcal{D}_n)) < \hat{V}(\mathcal{A}(\mathcal{D}_n)) - \kappa(r_n(\mathcal{D}_n))\right\}$$

and

$$\Lambda_{\mathcal{S}_r} = \left\{\mathcal{D}_n \in \Omega^n : V(\mathcal{A}(S_r)) < \hat{V}(\mathcal{A}(S_r); S_r) - \kappa(r)\right\},$$

where $\mathcal{S}_r$ is a subset of r indices $\{i_1, \ldots, i_r\}$ from $\{1, \ldots, n\}$ and $S_r = \mathcal{D}_n(\mathcal{S}_r)$. Intuitively, Λ is the set of "bad dataset" that leads to *overestimation* of RP, and $\Lambda_{\mathcal{S}_r}$ is the set of bad datasets due to the subset indexed by $\mathcal{S}_r$. Furthermore, consider a partition $\{\Omega_0, \ldots, \Omega_n\}$ of the sample space Ω^n, where $\Omega_r := \{\mathcal{D}_n \in \Omega^n : |\mathcal{I}_n(\mathcal{D}_n)| = r\}$ is a set of datasets that for which $\mathcal{I}_n$ returns a set with cardinality r. For each subset Ω_r, consider a further refinement indexed by all possible subsets $\mathcal{S}_r \subseteq \{1, \ldots, n\}$ with cardinality $|\mathcal{S}_r| = r$, $\Omega_{r,\mathcal{S}_r} \subseteq \Omega_r$, such that $\mathcal{D}_n \in \Omega_{r,\mathcal{S}_r}$ if and only if $\mathcal{I}_n(\mathcal{D}_n) = \mathcal{S}_r$. Obviously, it holds that $\Omega_r = \cup_{\mathcal{S}_r} \Omega_{r,\mathcal{S}_r}$, and $\Omega^n = \cup_{r=0}^n \cup_{\mathcal{S}_r} \Omega_{r,\mathcal{S}_r}$. We have that

$$\begin{aligned}
\Lambda &= \Omega^n \cap \Lambda \\
&= \bigcup_{r=0}^{n} \bigcup_{\mathcal{S}_r} \Omega_{r,\mathcal{S}_r} \cap \left\{V(\mathcal{A}(\mathcal{D}_n)) < \hat{V}(\mathcal{A}(\mathcal{D}_n)) - \kappa(r_n(\mathcal{D}_n))\right\} \\
&\overset{(i)}{=} \bigcup_{r=0}^{n} \bigcup_{\mathcal{S}_r} \Omega_{r,\mathcal{S}_r} \cap \left\{V(\mathcal{A}(S_r)) < \hat{V}(\mathcal{A}(S_r); S_r) - \kappa(r)\right\} \\
&\overset{(ii)}{=} \bigcup_{r=1}^{n} \bigcup_{\mathcal{S}_r} \Omega_{r,\mathcal{S}_r} \cap \Lambda_{\mathcal{S}_r},
\end{aligned} \tag{11}$$

where (i) is due to the definition of $\Omega_{r,\mathcal{S}_r}$, the property of $\mathcal{I}_n$, and Assumption 2, and in (ii) we removed the case for $r = 0$ since $\kappa(0) = 1$ and $\{\mathcal{D}_n \in \Omega^n : V(\mathcal{A}(\mathcal{D}_n)) < 0\} = \emptyset$.

Without loss of generality, we focus on a specific example with $\mathcal{S}_r = \{1, \ldots, r\}$. For any $\mathcal{D}_n = \{\xi_i\}_{i\in[n]} \in \Lambda_{\mathcal{S}_r}$, a necessary condition for $\mathcal{D}_n$ to belong to $\Omega_{r,\mathcal{S}_r} \cap \Lambda_{\mathcal{S}_r}$ is that $\mathcal{A}(\mathcal{D}_n(\mathcal{S}_r)) \in \hat{\Theta}_{\xi_i}$ for all $i \in [r + 1, n]$ due to the property of $\mathcal{I}_n$, so $\mathcal{A}(\mathcal{D}_n(\mathcal{S}_r)) \in \Theta_{\xi_i}$ by Assumption 1. Also, by the definition of $\Lambda_{\mathcal{S}_r}$, it holds that

$$\begin{aligned}
V(\mathcal{A}(\mathcal{D}_n(\mathcal{S}_r))) &= \mathbb{P}(\xi \in \Omega : \mathcal{A}(\mathcal{D}_n(\mathcal{S}_r)) \in \Theta_\xi) \\
&< \hat{V}(\mathcal{A}(\mathcal{D}_n(\mathcal{S}_r)); \mathcal{D}_n(\mathcal{S}_r)) - \kappa(r)
\end{aligned}$$

Therefore, by the independence of $\{\xi_i\}_{i\in[r+1,n]}$, we have

$$\begin{aligned}
&\mathbb{P}^{n-k}\left\{\{\xi_i\}_{i\in[r+1,n]} : \{\xi_i\}_{i\in[n]} \in \Omega_{r,\mathcal{S}_r} \cap \Lambda_{\mathcal{S}_r}\right\} \\
&\overset{(i)}{\leq} \mathbb{P}^{n-k}\left\{\{\xi_i\}_{i\in[r+1,n]} : \bigcap_{i=r+1}^{n} \mathcal{A}(\{\xi_i\}_{i\in[n]}) \in \Theta_{\xi_i}\right\} \\
&\overset{(ii)}{=} \prod_{i=r+1}^{n} \mathbb{P}\left\{\xi_i : \mathcal{A}(\{\xi_i\}_{i\in[n]}) \in \Theta_{\xi_i}\right\} \\
&\overset{(iii)}{\leq} \prod_{i=r+1}^{n} \left(\hat{V}(\mathcal{A}(\{\xi_i\}_{i\in[r]}); \{\xi_i\}_{i\in[r]}) - \kappa(r)\right) \\
&\overset{(iv)}{\leq} (1 - \kappa(r))^{n-r},
\end{aligned} \tag{12}$$

where (i) and (iii) follow by the necessary condition for $\{\xi_i\}_{i\in[n]} \in \Omega_{r,\mathcal{S}_r} \cap \Lambda_{\mathcal{S}_r}$, (ii) follows from the independence of ξ_i for all $i = r + 1, \ldots, n$, and (iv) is due to a simple upper bound that $\hat{V}(\theta; S_r) \leq 1$ by definition.

So far, we have conditioned on the observations ξ_i for $i = 1, \ldots, r$. Integrating over these observations gives

$$\begin{aligned}
\mathbb{P}^n(\Omega_{r,\mathcal{S}_r} \cap \Gamma_{\mathcal{S}_r}) &\leq \int (1 - \kappa(r))^{n-r} \, \mathbb{P}^r(d\xi_1, \ldots, d\xi_r)) \\
&\leq (1 - \kappa(r))^{n-r}.
\end{aligned}$$

Recall that $\mathcal{S}_r = \{1, \ldots, r\}$ is chosen as an example, and the above holds for any $\mathcal{S}_r$ and any $r = 0, \ldots, n - 1$. Therefore, by a simple union bound, we have that

$$\begin{aligned}
&\mathbb{P}^n\left(V(\mathcal{A}(\mathcal{D}_n)) < \hat{V}(\mathcal{A}(\mathcal{D}_n)) - \kappa(r_n(\mathcal{D}_n))\right) \overset{(i)}{=} \mathbb{P}^n(\Lambda) \\
&\overset{(ii)}{=} \mathbb{P}^n\left(\bigcup_{r=1}^{n} \bigcup_{\mathcal{S}_r} \Omega_{r,\mathcal{S}_r} \cap \Lambda_{\mathcal{S}_r}\right) \leq \sum_{r=1}^{n} \sum_{\mathcal{S}_r} (1 - \kappa(r))^{n-r} \\
&\overset{(iii)}{=} \sum_{r=1}^{n} \binom{n}{r} (1 - \kappa(r))^{n-r},
\end{aligned} \tag{13}$$

where (i) is by definition of Λ, (ii) is by (11), and (iii) is because there are $\binom{n}{r}$ choices of $\mathcal{S}_r$.

C. Experimental details

Here is a description of model specifications:

- vanilla NODE [7]: We instantiate $f_\theta(z)$ as a 3-layer neural network with 784 hidden units in each hidden layer and ReLU activation. The model is trained with clean data using cross-entropy loss.
- AT-NODE: the NODE shares the same architecture as the vanilla NODE. The adversarial example is generated by L_∞ PGD attack. Following standard adversarial training process, the parameter is trained with a cross-entropy loss on both adversarial and clean samples.
- B-NODE: the proposed method trained with (8), which shares the same architecture as the vanilla NODE and the same data augmentation strategy as AT-NODE.
- ODE-TRADES [24]: the NODE shares the same architecture as the vanilla NODE. We follow the training procedure proposed in [24], which also includes data augmentation with adversarial examples generated for randomly perturbed data with L_∞ PGD attack.

For forward passes, we use Euler's method and set the integration time $T = 1$ and stepsize $dt = 0.001$; for backward passes, the adjoint sensitivity method is implemented.

Experiment on MNIST. For the MNIST data, we vectorize the original image (of size 28 by 28 pixels) to a 784-dimensional vector, which is then projected into a 25-dimensional vector through an affine transformation (i.e., the input layer of NODE is $\phi(x) = Ax$ for some matrix A). For the training of AT-NODE, B-NODE, and ODE-TRADES, the adversarial example is generated by 40 steps L_∞ PGD attack (with attack radius $\varepsilon = 0.05$).

Experiment on FashionMNIST. For the FashionMNIST dataset, we adopted the convolutional architecture proposed in [10] for NODE. To generate adversarial samples for data augmentation, we employed a PGD attack with $\varepsilon = 8/255$ and 40 steps. For ODE-TRADES baseline, the training data was first perturbed by Gaussian noise with a standard deviation of $10/255$, and then we used the PGD attack method with $\varepsilon = 8/255$ and 40 steps to generate the adversarial samples for augmentation.

Experiment on CIFAR10. For the CIFAR10 dataset, we followed the practice proposed in [7] and utilized a pre-trained CNN as the feature extractor. Specifically, for both our method and the baselines, we used the ResNet18 model provided by `https://github.com/huyvnphan/PyTorch_CIFAR10`, and employed the outputs (10-dimensional feature) of this model as input for NODEs. It's important to note that the feature extractor was fixed during the training of Neural ODE. The dynamics of the NODE have a fully connected neural network architecture with 3 layers, an input and output dimension of 10, a hidden dimension of 25, and ReLU activation. All three models AT-NODE, B-NODE and ODE-TRADES are trained with clean data augmented with adversarial examples, which are generated by 50 steps L_∞ norm PGD attack, whose magnitude ε is 3/255.

D. Details of attack models

For PGD attacks, we implement the method provided in [18] with 40 steps and varying attack radius (we use 50 steps for CIFAR10 experiment). For AutoPGD attacks, we refer to [8] and use L_∞ norm attack with a cross-entropy loss function and an update step size of 0.75. For Square Attack, we follow [2] with the L_∞ norm. This attack is generated using 5000 queries and employs a margin loss function.

E. Learning barrier function and an additional example

In our method, the barrier function certificate is learned (as reflected in our title). We choose a natural candidate for $h_y(z) = [Wz]_y - \max_{k \neq y}[Wz]_k$, where W coincides with the final layer of NODE and is learned by Optimization (8).

To further clarify on this important point: For classification problems, the correctness condition is given by

$$\nu_y \geq \nu_{y'}, \forall y' \neq y,$$

where $\nu = Wz(T)$ is the output of a NODE and y is the true label. Thus, the safe region is given by $\mathcal{C}_y = \{z : [Wz]_y \geq [Wz]_{y'}, \forall y' \neq y\}$. This motivates our definition of h_y, because $\mathcal{C}_y = \{z : h_y(z) \geq 0\}$ by definition.

It's worth noting that a more complex certificate function, such as $h_y(z) = [NN_\omega(z)]y - \max k \neq y[NN_\omega(z)]_k$, where ω represents the weights of the neural network. Here, $NN_\omega(z)$ represents a neural network with input z and output of the same dimension as the number of classes. In this case, it is necessary to ensure that $\{z : h_y(z) \geq 0\} \subseteq \mathcal{C}_y$ as required by Theorem 1 (note that the relationship is equality if we pass the output state of NODE $z(T)$ through NN instead of the linear layer parametrized by W). However, we have found that the simple construct presented in the paper is sufficient to improve performance.

F. Convergence of PGD attack

In our MNIST experiment, we reported the results against a 40-step PGD attack following the practice of [18]. To demonstrate that the PGD attack has converged, we compared the performance of our models against a 50-step PGD attack, as shown in Table VIII. As observed, there was no significant difference in accuracy results between the two attacks. Therefore, we concluded that a 40-step PGD attack was an effective adversarial perturbation for our models.

TABLE VIII: Robustness against PGD attack with 50 steps.

	PGD-0.01	PGD-0.02	PGD-0.03	PGD-0.04	PGD-0.05
vanilla NODE	85.25±2.27	80.05±1.64	70.80±2.21	59.55±2.36	46.02±3.41
AT-NODE	91.40±0.95	90.26±1.87	85.86±1.44	82.46±2.19	79.16±1.75
ODE-TRADES	91.56±1.22	89.36±1.30	85.24±1.72	81.84±2.14	78.47±2.04
B-NODE	**93.10±1.33**	**91.30±2.01**	**88.50±1.87**	**86.55±1.74**	**82.89±2.09**

G. Ablation analysis details

For the ablation analysis, we considered four different models. Model 1 is a vanilla NODE trained with clean data using cross-entropy loss. Model 2 is a NODE trained with clean data and barrier function regularization (8). Model 3 is a vanilla NODE trained with clean data augmented with adversarial data using cross-entropy loss. Model 4 is our proposed method, which uses clean data augmented with adversarial data trained with (8).

H. Sensitivity analysis

The influence of the values of λ_i is shown in Fig. 2, where we vary the values of λ_i by choosing from the set $V = \{0.1, 0.5, 1, 5\}$. The figure shows the accuracy of against L_∞ PGD attacks with $\varepsilon = 0.05$ and 40 steps on MNIST. In general, increasing the values of λ_i results in more robust performance.

I. Ablation analysis

We conducted an experiment to analyze the impact of learning-based barrier functions and data augmentation on the robustness of four different models (see details in [22, Appendix]). As shown in Table IX, barrier function without data augmentation can already improve robustness, sometimes by 15% (in the case of PGD-0.05). It is remarkable to see improvement in all cases of attacks, indicating some level

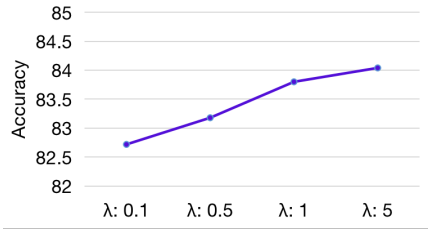

Fig. 2: Sensitivity analysis for regularization coefficient λ_i corresponding to $\mathcal{L}_i(\theta)$ in (8).

of "universal robustness." However, the use of the barrier function in conjunction with data augmentation yields the most robust performance.

TABLE IX: Ablation analysis for models with and without barrier function loss or data augmentation on MNIST dataset.

	PGD-0.01	PGD-0.02	PGD-0.03	PGD-0.04	PGD-0.05
vanilla NODE	86.14±1.33	79.26±1.80	71.04±2.01	60.36±2.34	47.40±2.11
B-NODE w/o AT	88.62±1.40	83.82±1.10	79.00±1.39	71.22±2.23	61.46±1.64
AT-NODE	91.76±1.16	89.34±1.14	85.80±1.94	82.94±1.64	78.38±1.67
B-NODE w/ AT	92.62±0.87	91.52±1.217	89.16±1.25	86.36±1.78	83.18±1.75

J. Experiment with convolutional NODE

Our approach centers on designing the training loss function (8) and is not limited to specific neural network architectures. Given the similarity of the forward propagation process between fully connected and convolutional layers, our method can also be applied to convolutional NODEs (C-NODEs). Specifically, using the implementation provided in `https://github.com/EmilienDupont/augmented-neural-odes`, where the C-NODE uses 2D matrices with the same channels as the input images as hidden states, we conducted additional experiments on the MNIST dataset. The results demonstrate that our method can enhance the robustness of C-NODEs as well (Table X). Here is the description for C-NODEs.

- C-NODE: The function $f_\theta(z)$ is generated by three 2D convolution layers, with shapes given as [[1,5,1,1,0], [5,5,1,1,1], [5,1,3,1,0]]. The parameters of each layer denote the input channels, output channels, kernel size, stride, and padding. The model is trained with cross-entropy.
- Barrier C-NODE: shares the same framework with C-ODE. Trained with loss function in (8).

TABLE X: Performance improvement on C-NODE with barrier function on MNIST.

	PGD-0.01	PGD-0.02	PGD-0.03	PGD-0.04	PGD-0.05
C-NODE	85.50±2.29	78.49±2.57	73.70±3.06	67.40±3.32	63.50±5.14
Barrier C-NODE	**88.50±3.74**	**83.10±3.96**	**80.29±2.28**	**74.70±3.84**	**70.59±4.60**