AntiNODE: Evaluating Efficiency Robustness of Neural ODEs

Mirazul Haque[1], Simin Chen[1], Wasif Haque[3], Cong Liu[2], Wei Yang[1]

[1]University of Texas at Dallas, USA, [2]University of California, Riverside, USA, [3]Argo Data

mirazul.haque@utdallas.edu

Abstract

Recently, Neural ODE (Ordinary Differential Equation) models have been proposed, which use ordinary differential equation solving to predict the output of neural networks. Due to Neural ODE models' noticeably lower parameter usage compared to traditional Deep Neural Networks (DNN) and higher robustness against gradient-based attacks, they are being adopted in many type of real-time applications. For real-time applications, response-time (latency) has paramount importance due to the convenience of the user. Through our observation, we find that the latency during Neural ODE inference can be highly dynamic and sometimes detrimental to the system due to the adaptive nature of the ODE solvers. Because of that reason, understanding and evaluating efficiency robustness of Neural ODE models is needed, which has not received much attention yet. However, evaluating efficiency robustness of any model is dependent on the relationship between input and latency, which has not been defined yet for Neural ODE models. In this work, we first formulate the relationship between input and dynamic latency consumption of Neural ODEs. Based on the formulation, we propose `AntiNODE`*, which generates latency-surging adversarial inputs for Neural ODEs by increasing the computations in Neural ODEs. We evaluate* `AntiNODE` *on two popular datasets and three ODE solvers on both hardware dependent and independent metrics. Results show that the adversarial inputs generated by* `AntiNODE` *can decrease up to 335% efficiency during inference. Our evaluation also shows that the generated adversarial inputs are transferable across multiple solvers and multiple architectures, which indicates the feasibility of black-box attack.*

1. Introduction

Deep Neural Networks (DNNs) have shown great potential in many challenging tasks. Handling tasks with higher complexity requires a rapidly increasing number of DNN parameters. To address this issue, recently, researchers simulated the solver of ordinary differential equation (ODE) and proposed Neural ODE networks [10]. Because of Neural ODE's parameter efficiency and invertibility, Neural ODE has increased attention of the research community recently. Neural ODE does not store any intermediate quantities of the forward pass and enables training DNNs with small and constant memory cost. Recent studies [10] have shown that neural ODEs often perform better than traditional DNNs for irregularly sampled time-series data. Other than that, recent studies have shown that Neural ODEs can be used for various types of applications like online prediction [41], video generation [43], robotic manipulator control [40], disease progression modeling [38], node classification [44] etc.

Because of the wide deploy-ability, the robustness of the Neural ODE is needed to be evaluated. Through recent study [8], it has been noticed that Neural ODE is more robust against specific gradient-based accuracy attacks. However, the evaluation of latency-based efficiency robustness of Neural ODE's is also important. For example, a visually impaired person relies on a mobile application for navigation that uses Neural ODE, while the response time is significantly higher for specific inputs. This can lead to fatal consequences.

To evaluate efficiency robustness of Neural ODE's, we conduct a preliminary study (Section 3). We observe that Neural ODEs perform adaptive computations leading to highly volatile latency. For example, through Figure 1, we can observe that to classify MNIST data, different latency occurs for different inputs in a Neural ODE model. Also, in this work, we have found out that it is feasible to synthesize inputs whose latency is 335% higher than benign inputs.

Such inputs can lead to disastrous consequences, especially for real-time scenarios.

To avoid such fatal consequences, it is critical to understand the *efficiency robustness* [25] in Neural ODE networks. To understand the efficiency robustness of Neural ODEs, the relationship between input and latency of Neural ODEs needs to be defined. Recent works [25, 30] have proposed efficiency attack of Adaptive Neural Networks (AdNNs) [31, 20, 36, 51, 23, 50, 55] by proposing intermediate output-based loss function. Based on intermediate outputs, the AdNNs deactivate certain computations of DNNs to improve latency. However, the relationship between input and latency of Neural ODEs is different from the relationship in AdNNs. For Neural ODE, no component is deactivated or activated during inference. Therefore, the same techniques used for evaluating efficiency robustness in AdNNs can not be used for Neural ODEs.

To make progress in understanding the efficiency robustness of Neural ODEs, we draw a key finding: the latency in Neural ODE is correlated with the adaptive number of iterations needed by ODE solvers to predict the output. ODE solvers use an iterative way to approximate a function, where the number of iterations is adaptive. More number of iterations means higher latency of the model. However, there is no direct relation between the number of iterations of the ODE solver and the input because the gradient is non-existent between them. To address this issue, we use the adaptive step-size as the intermediate proxy to relate the iterations of the ODE solver to the input (Section 4). The high latency inputs are then synthesized based on step sizes.

Based on our findings, we propose `AntiNODE`, a white-box adversarial attack that uses step-size of the ODE solvers to generate adversarial inputs. Based on the restriction of the adversarial perturbations (Section 4), we develop two techniques, namely restricted efficiency attack and unrestricted efficiency attack. On the one hand, restricted efficiency attack evaluates efficiency robustness where generated adversarial inputs are semantically meaningful to the Neural ODE model. On the other hand, unrestricted efficiency attack evaluates worst-case efficiency robustness where each adversarial input maximizes the latency for each target ODE solver. To the best of our knowledge, this is the first work to generate efficiency-based adversarial inputs against Neural ODEs.

We evaluate `AntiNODE` on mainly two criteria: effectiveness and transferability using the CIFAR-10 and MNIST datasets [34, 16]. We evaluated `AntiNODE` on two popular ODE solvers: Dopri5 [18] and Adaptive Heun [48]. We evaluate the effectiveness of `AntiNODE` against natural corruptions [29] and PGD attack [39]. We observed that `AntiNODE` generated adversarial inputs can increase up to 335 % latency than the average latency of original benign inputs. Also, we noticed that transferability is feasible between two Neural ODEs differentiated by ODE solver or model architecture. Our paper makes the following contributions:

- **Problem Characterization.** Our work is the first work to characterize the latency surge vulnerability in the neural ODE networks. Such vulnerability can be generalized to any other neural networks that apply dynamic computations for different inputs.
- **Approach.** Our work is the first attempt to formulate the relationship between input and latency of Neural ODE models. Based on the formulation, our work proposes a novel loss function to generate adversarial inputs. Also, through evaluating the transferabilty, we discuss the feasibility of black-box attack.
- **Evaluation.** We assess our method using two criteria across two ODE solvers and two datasets. The findings reveal that existing accuracy-based attacks cannot increase latency in Neural ODE models. On the contrary, adversarial inputs generated by `AntiNODE` can raise latency by up to 335%.

2. Background and Related Works

. Ordinary Differential Equations Solver

The Runge-Kutta method ([46, 35]) is an ODE solver which solves ordinary differential equations through approximation. First-order differential equation given by,

$$\frac{dy(t)}{dt} = y^{'}(t) = f(y(t), t)$$

with $y(t_0) = y_0$ Here y is an function of time t and y^* is the value of y at $t = 0$. The objective of the ODE solver is to approximate the function y at certain point $t = t_0$ using multiple iterations. Four slope approximations k_1, k_2, k_3, k_4 are used to estimate approximate value of y (y^*) at $t = t_0$. Final estimate of $y^*(t_0 + h)$ can be represented as,

$$y^*(t_0 + h) = y^*(t_0) + (\frac{1}{6}.k_1 + \frac{1}{3}.k_2 + \frac{1}{3}.k_3 + \frac{1}{6}.k_4).h$$

Here, h is the step size and k values are the intermediate values. This is called fourth order Runge Kutta Method, because the local error (approximation error at a particular time) for step-size h is $O(h^4)$. For better approximation of function, multiple works [18, 48] have been proposed to use adaptive step size.

. Neural Ordinary Differential Equations Networks

Neural Ordinary Differential Equations Networks (Neural ODE) [10] have been successful in attaining accuracy close to the state of the art DNN techniques but with lesser memory consumption. Neural ODEs incorporate Ordinary

Differential Equations solvers into Neural Network architectures. Models such as residual networks and recurrent neural network decoders create complicated transformations by devising a sequence of transformations to a hidden state:

$$h_{t+1} = h_t + f(h_t, \theta_t)$$

Operation of a residual block can be interpreted as the discrete approximation of an ODE where the discretization step value is one. In a neural ODE, the discretization step is set to zero and the relation between input, and output is characterized by the following set of equations:

$$\frac{\mathrm{d}h(t)}{\mathrm{d}t} = f(h(t), t, \theta), \qquad h(0) = h_{in}, \qquad h_{out} = h(T)$$

Solving $h(T)$ provides the output and ODE solvers can be used for that purpose.

The objective of ODE solver is to approximate a function given the differential equation of the function. To approximate a function, on specific points (t), function slope value is calculated using the differential equation and the function is approximated. The difference between t values on which the slope is calculated is called step-size. If two t values are nearer (low step-size), more number of iterations (slope calculation) is needed for function approximation.

Significant amount of research has been done on Neural ODEs based on training optimization, approximation capabilities, and generalization. First, Chen *et al.* [10] propose to compute gradients using the adjoint sensitivity method, to create a memory efficient training method. In this training method, there is no need to store any intermediate quantities of the forward pass. Proposed work by Quanglino *et al.* [45] expresses the Neural ODE dynamics as truncated series of Legendre polynomials and accelerate the model. Dupont *et al.* [19] explores the limitations in approximation capabilities of neural ODEs because of the preserving of input topology. Recent work by Yan *et al.* [53] explore the robustness of Neural ODEs against Neural ODEs and propose TisODE to increase the robustness of Neural ODEs. However, no other work has focused on the efficiency robustness perspective or Neural ODEs, and to our knowledge, this is the first work in that direction.

. Adversarial Examples.

Adversarial Examples are the inputs that are used to test the robustness of machine learning models. In earlier works by [15, 37, 17], 'good word attacks' or spelling modifications have long been used to bypass the spam filters. More recently, Szegedy *et al.* [49] and Goodfellow *et al.* [22] propose adversarial attacks on deep computer vision models. With a similar approach, adversarial attacks have been extended to various fields like text and speech processing [6, 33], and graph models [56, 3]. Recent works [25, 30, 26, 24, 27, 12, 14, 13, 11, 54] have proposed white-box testing methods for Adaptive Neural Networks. However, as mentioned in the introduction, existing attacks can not be used to increase latency of Neural ODEs. Also, Haque *et al.* [28] propose a black-box testing technique for AdNNs by creating an energy estimator model. However, for training the estimator model, target models need to show significant adaptive computation for different ordinary data points, which is not feasible for Neural ODE. For Neural ODE, for ordinary inputs, significant adaptive computation can not be shown.

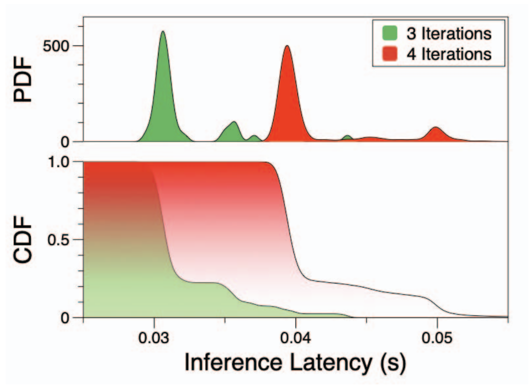

Figure 1: Difference in latency with different number of iterations

3. Adaptive Nature of Neural ODE

In this section, we discuss the adaptive latency of Neural ODEs. As discussed in Section 2, ODE solvers use iterative approximation to calculate the function value at a certain point. If the number of iterations is increased, the function f is needed to be calculated for a higher number of times, increasing the latency of the process. The number of iterations can be increased by decreasing the step-size, which we discuss in Section 4.

We also have investigated the adaptive nature of Neural ODEs through a preliminary study. We use a Neural ODE model from existing work [10], which is trained for MNIST [16] data. For ODE solver, we follow the author [10] and use dopri5 [18] as ODE solver. We then feed the test data from MNIST into the Neural ODE model for classification.

Figure 1 describes our findings. From the results, we have noticed that ODE solver takes a different number of iterations (3 and 4 in this experiment) to approximate a function. It can be noticed from the figure that when 3 iterations are performed, the probability is higher that the inference latency is 0.03s. Similarly, if the number of iterations is 4, it is more probable that latency during inference is 0.04s.

Therefore, when the number of iterations is 3, average latency drops more than 0.01s. However, the range of inference latency is limited is low for in-distribution data.

4. AntiNODE

. Problem Formulation

The objective of `AntiNODE` is to generate inputs to decrease ODE network efficiency and increase latency. Based on the restriction of the attackers [4, 47], we consider two attack scenarios. First is the restricted efficiency attack, which seeks to perturb a benign image with human-unnoticeable perturbations to increase the latency of Neural ODEs. The imperceptible efficiency attack should satisfy the following three properties: *(i)* the generated inputs should increase the latency of the victim Neural ODE. *(ii)* the perturbed inputs should be imperceptible to human beings. *(iii)* the generated inputs should be realistic. In other words, generated inputs should maintain the domain-specific constraints. As indicated in Eq 1, we express the above properties as an optimization problem.

$$\begin{aligned} \Delta(x) &= \text{argmax}_\delta \quad \text{Latency}_{\mathcal{F}}(x+\delta) \\ s.t. \quad & ||\delta|| \le \epsilon \ \wedge \ ||x+\delta|| \in [0,1]^n \end{aligned} \tag{1}$$

Here, x is an given benign input, $\text{Latency}_{\mathcal{F}}(\cdot)$ measures the latency of ODE solver $\mathcal{F}$, δ is the perturbation under optimization, ϵ is the imperceptible constraint, Δ is the generated adversarial perturbation. Another attack scenario is the unrestricted efficiency attack [47], which does not consider the imperceptible constraints and aims to understand the worst case latency of Neural ODEs. We formulate the objective of unrestricted efficiency attack as Eq 2.

$$\begin{aligned} \delta &= \text{argmax}_\delta \quad \text{Latency}_{\mathcal{F}}(x+\delta) \\ s.t. \quad & \qquad ||x+\delta|| \in [0,1]^n \end{aligned} \tag{2}$$

We follow existing work [7] and consider the white-box configuration in this study to fully highlight the latency surge vulnerabilities in neural ODE networks. Furthermore, we assess the transferability of adversarial examples in §5 to demonstrate the feasibility of launching black-box attacks.

. Design Overview

For detailed explanation, we mainly focus on restricted efficiency attack scenario. The attack overview of `AntiNODE` is explained through Fig. 2. `AntiNODE` first transforms the initialized adversarial example into optimization space (denoted as $\hat{x} = \text{T}(x+\delta)$). After that, `AntiNODE` iteratively performs the following three steps: ① First, we get a differentiable proxy to the latency of the neural ODE networks; ② Next, we compute the importance score for ODE solver steps based on the which step's optimization is more important than other steps; and ③ We construct the objective function based on the added perturbation and differentiable proxy of the latency and update the optimization variable $\hat{x}$. Finally, `AntiNODE` transforms $\hat{x}$ from the continuous optimization space back into the input domain to generate the adversarial examples. The adversarial examples generated will significantly raise the victim Neural ODE networks' latency.

. Detailed Design

Continuous Space Transformation. To ensure the generated adversarial example be a valid input, we need to ensure the adversarial example satisfies the validity constrains in Eq. (1) and (2). Such constraints are known as box constraints in the optimization theory. To satisfy the constraints, we introduce a continuous latent variable $\hat{x}$, which can be transformed from x and δ, and optimize over $\hat{x}$ instead of optimizing δ. In detail, we introduce the transformation as Eq. 3.

$$\begin{aligned} \hat{x} &= \text{T}(x+\delta) = atanh(2 \times (\delta + x) - 1) \\ x + \delta &= \text{T}^{-1}(\hat{x}) = \frac{1}{2}(tanh(\hat{x}) + 1) \end{aligned} \tag{3}$$

Here $tanh(x) = \frac{e^x - x^{-x}}{e^x + x^{-x}}$, and $atanh(x) = \frac{1}{2} ln(\frac{1+x}{1-x})$. The function $tanh(\cdot)$ ranges form -1 to 1, thus, $x + \delta$ will always locate in the range of $[0, 1]$ and satisfy our validity constraints. Moreover, our continuous space transformation will enable our optimization process on $\hat{x}$ to be performed in a continuous space and avoid the disadvantage of discontinuous in gradient clip methods.

Differential Latency. As mentioned earlier, the goal of the adversary is to increase the latency of the Neural ODE networks. However, the latency of Neural ODE during inference is non-differentiable, which implies it is challenging to apply the gradient-based approach to generate the optimal adversarial perturbations. To address this challenge, we propose a differentiable function to approximate the latency of Neural ODE in hanging an input. As mentioned in the background, Neural ODE will iteratively compute the function gradient value on different point t until the function is not approximated. Based on such natural property of Neural ODE networks, our intuition is that we can use the number of iterations to approximate the neural ODE networks latency. Furthermore, the number of iterations is determined by the *step_size* H of each iteration. Thus, we will collect the *step_size* ($H(x+\delta)$) of the neural ODE networks in handling current inputs $x + \delta$ and use it to approximate neural ODE networks latency. Because the *step_size* $H(x+\delta)$ is computed form the current inputs $x + \delta$, $H(x+\delta)$ is differentiable in terms of $x + \delta$.

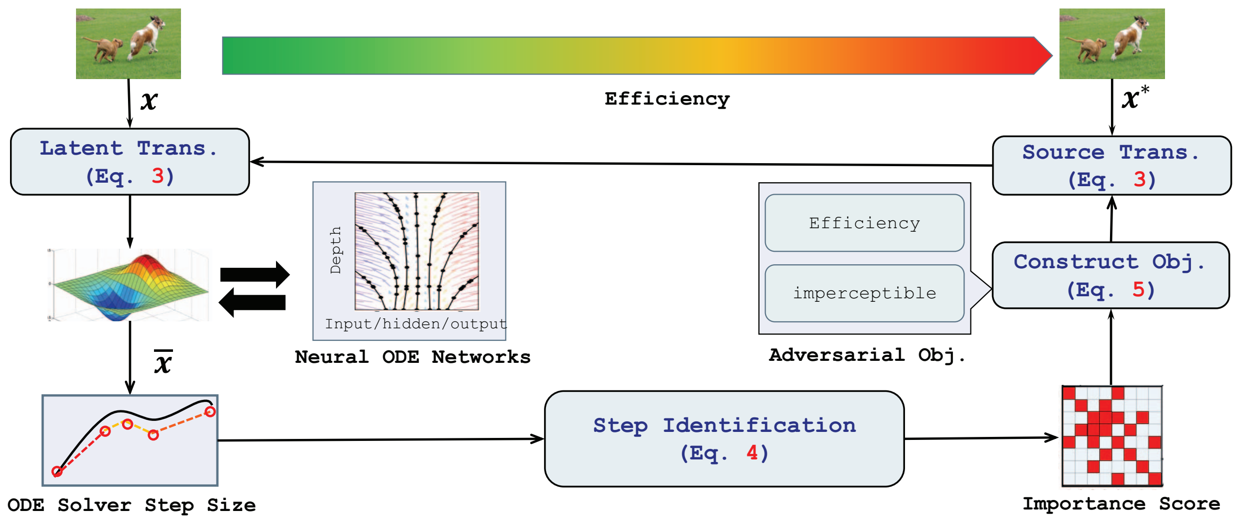

Figure 2: Overview of AntiNODE

Algorithm 1 AntiNODE

```
Input: Benign input $x$, victim neural ODE networks $\mathcal{F}(\cdot)$, maximum iteration number $T$.
Output: Latency-based adversarial example $x^*$
 2: Initialize $\delta, x^*, max_N$
 3: $\hat{x} \leftarrow T(\delta + x)$
 4: while $iter < T$ do
 5:     $H$ =CollectStepSize $(\mathcal{F}, x + \delta)$
 6:     $H_{avg} \leftarrow Importance(H)$
 7:     $\mathcal{L}_d \leftarrow Distance(\mathrm{T}^{-1}(\hat{x}), x)$
 8:     Compute $\mathcal{L}_{adv}$ based on Eq. 5
 9:     $\hat{x} \leftarrow \hat{x} + \frac{\partial \mathcal{L}_{adv}}{\partial \hat{x}}$
10:     $x_{current} \leftarrow \mathrm{T}^{-1}(\hat{x})$
11:     $N \leftarrow$ GetSteps$(x_{current})$
12:     $max_N, x^*$ = Update$(max_N, N, x_{current})$
13: end while
14: Return $\mathrm{T}^{-1}(x^*)$
```

Importance Score Computation. After we collect the *step_size* that the neural ODE networks require to handle an input, we need to find the approximate relation between *step_size* H and latency. We know that the neural ODE's latency is positively correlated to its ODE solvers' iterations. To increase the number of iterations, we need to make sure that the *step_size* values are decreased. As the *step_size* values are decreased, the ODE solver would need more steps to approximate the function, thus increasing the number of iterations. However,all the H values do not have the same importance. Intuitively, if a step-size is significantly larger than the other step-size, more weight should be put on the larger step-size during optimization process. Hence, we propose to use the exponential value of the step-sizes, such that, larger step-size has higher importance score during the optimization process. We then normalize the average *step_size* based on the importance scores. In detail, we compute the importance score and normalized average *step_size* using Eq. 4.

$$w_i = \frac{e^{H_i}}{\sum_{j=1}^{N} e^{H_j}} \qquad H_{avg}(x) = \frac{\sum_{j=1}^{N} w_i}{N} \tag{4}$$

Attack Objective Function. Recall that our efficiency attack requires adversarial examples to raise latency while remaining unnoticeable to humans.

$$\begin{aligned}\mathcal{L}_{latency} &= \text{minimize}_{\hat{x}} H_{avg}(x)\\ \mathcal{L}_{per} &= \text{minimize}_{\hat{x}} \|\mathrm{T}^{-1}(\hat{x}) - x\|\\ \mathcal{L}_{adv} &= \text{minimize}_{\hat{x}} \mathcal{L}_{latency} + \beta \mathcal{L}_{per}\end{aligned} \tag{5}$$

Our objective function can be found in Eq 5, where we have two objectives, the first is to increase latency ($\mathcal{L}_{latency}$) and the other one is to keep the adversarial examples imperceptible ($\mathcal{L}_{per}$). And our final adversarial objective is $\mathcal{L}_{adv}$, which combines both two aforementioned objectives. For the unrestricted efficiency attack, we set $\beta = 0$ because we do not have the imperceptible constraints for this attack.

Our general attack algorithm is shown in Alg. 1.

5. Evaluation

. Experimental Setup

Datasets and Models. For evaluation, CIFAR-10 dataset [34] and MNIST dataset [16] have been used for the training of the Neural ODE model. CIFAR-10 is an RGB image dataset where MNIST is a grey image dataset. Therefore, we can analyze the impact of testing inputs on both types of data. We use ODENet Convolutional Neural Network (CNN) models proposed by [10] as the trained

Neural ODE models. The main difference between model architectures used for CIFAR-10 and MNIST is in the input number of channels. We evaluate the performance of our techniques on three popular different adaptive ode solvers, `Dopri5` [18], `AdaptiveHeun` [48] and, `Bosh3` [2].

Baselines. As far as we know, we are the first to attack the Neural ODEs in terms of latency. Thus, there is no existing comparison baselines. To show that existing accuracy-based adversarial examples can not increase the latency in neural ODEs, we compare our technique with natural corruption techniques [29] and PGD [39]. These techniques are commonly used [52, 21, 42] to evaluate the accuracy robustness of neural networks. As the MNIST dataset consists of grey images, all types of corruptions can not be applied to the images. Therefore, for MNIST dataset, we use random Gaussian noise [9] and Salt and Pepper noise [32] as baseline. These noises have been used previously [5] on MNIST dataset to generate noisy inputs.

Metric. We examine two metrics to reflect the neural ODEs' computational costs in order to measure the effectiveness of tool in increasing the victim neural ODEs' computational costs. The first is the number of iterations used by the ODE solver (hardware independent metric), and the second is the latency of Neural ODE networks (hardware dependent metric) in handling an input. We first measure the absolute computational costs (Abs.) of the benign inputs and then generate adversarial examples. After we compute the computational cost increments (Inc.). For transferability evaluation, we consider two metrics, Input Transferability Percentage (ITP) and Effectiveness Transferability Percentage (ETP), used by Haque et al. [28] to evaluate energy testing transferability.

$$\text{ITP} = \frac{\sum_{x \in \mathcal{X}} \mathbb{I}(\text{Latency}(x+\delta) > \text{Latency}(x))}{N} * 100\%$$

$$ETP = \frac{\sum_{x \in \mathcal{X}} \text{Latency}(x+\delta) - \text{Latency}(x)}{\text{Latency}(x)} * 100\%$$

Here. x represents benign input and $x + \delta$ represents adversarial input generated on base model [28]. ITP evaluates the percentage of examples that can increase the latency in the target model. The other metric ETP evaluates the average percentage elatency increase in the target model.

Implementation Details. For restricted attack we use multiple β values to generate test inputs. For MNIST dataset, we have experimented with three β values: 10, 100, and 1000. Where for CIFAR-10, we have used 10, 100, 1000, and 10000 as β values. For reporting the effectiveness, transferability and quality, we report the result for β values for which the effectiveness of `AntiNODE` is highest. We have used 2000 iterations for restricted attack.

. Effectiveness

We have measured the effectiveness of `AntiNODE` generated inputs by measuring the average latency (in Seconds) induced by the inputs during the inference and comparing it with mean latency of the original seed inputs and inputs generated by other baseline techniques. We also measure the mean number of solver iterations caused by each technique. We measure the effectiveness of our approach for CIFAR-10 and MNIST datasets. For CIFAR-10 dataset, we use images generated by common corruptions as one of the baselines. For comparison, we use the two best performing corruptions in terms of average latency. For MNIST dataset, we use Gaussian noise and Salt and Pepper noise as the corruption baseline. Other than the corruptions, for both datasets, we also use PGD attack as baseline.

Table 1 show the effectiveness of `AntiNODE` on Dopri5, Adaptive Heun and Bosh3 solvers for both datasets. It can be observed that, for all scenarios, `AntiNODE` is able to generate higher latency examples than baseline methods. For unrestricted attack inputs, `AntiNODE` is able to generate higher latency inputs than the restricted attack inputs. Specifically for Adaptive Heun solver, latency increase caused by unrestricted and restricted attack inputs are significantly higher than latency increase in other two solvers. Both corruption based baseline and adversarial attack based baseline (PGD attack) do not perform well in terms of increasing latency. For PGD attack, the loss function is focused only on accuracy, and this does not impact the latency. Hence, we can conclude that latency-specific loss function is more successful in increasing the latency of models.

. Transferability

In this section, we evaluate the transferability of the restricted attack-generated inputs on different ODE solvers and on different network architectures. Transferability is evaluated to know if the adversarial inputs can be effective in a black-box scenario. To evaluate the transferability, two models are considered: Base Model and Target Model. First, the adversarial inputs are generated on Base Model and the generated inputs are fed to Target Model. If the adversarial inputs are effective on target model too, the adversarial input is transferable. Here, we consider transferability with respect to both architecture and solver. To evaluate the latency-based transferability of the adversarial inputs, we will measure two parameters used by Haque *et al.* [28]: 1. What percentage of adversarial inputs can increase the latency for the other solver/architecture? This is called Input Transferability Percentage (ITP) 2. What is the average percentage increase in latency enforced by adversarial inputs in other solver/architecture? This is called Effectiveness Transferability Percentage (ETP).

To measure the transferability, we have selected 1000

Table 1: **Mean latency and number of iterations of the models against `AntiNODE` and baseline techniques.** $Corr$ **represents the corruptions,** PGD **is the accuracy-based attack baseline,** $Unres$ **and** Res **represents unrestricted and restricted attack respectively. The results provide both absolute values and the percentage increase because of the technique. All the experiments are performed on CIFAR-10 and MNIST datasets and three aforementioned solvers.**

Metric	Dataset	ODE Solver	seed	Corr1		Corr2		PGD		Unres (Ours)		Res (Ours)	
			Abs.	Abs.	Inc	Abs.	Inc	Abs.	Inc	Abs.	Inc	Abs.	Inc
# of Iterations	CIFAR10	Dopri5	4	4.25	6.25%	4.25	6.25%	4.05	1.25%	**5.7**	**42.5%**	**5.5**	**37.5%**
		AdapHeun	45	45.5	1.%	45.5	1.1%	47.1	4.6%	**101**	**124%**	**73.28**	**62.8%**
		Bosh3	8.3	9.8	18%	9.14	10%	8.5	2.4%	**13**	**58.5%**	**10.9**	**32.9%**
	MNIST	Dopri5	4	4.14	3%	4.08	2%	4.05	1.25%	**6**	**50%**	**5.5**	**37.5%**
		AdapHeun	32	32.12	0.3%	31.9	-0.3%	32.3	0.9%	**89**	**178%**	**43.28**	**35.2%**
		Bosh3	7.1	7.1	0%	7.1	0%	7	-1%	**13**	**84.5%**	**9.9**	**35.2%**
Latency	CIFAR10	Dopri5	0.035s	0.037s	4.4%	0.037s	4.4%	0.036s	2%	**0.055s**	**57.1%**	**0.05s**	**42.8%**
		AdapHeun	0.145s	0.1475s	1.7%	0.1475s	1.7%	0.155s	7.2%	**0.425s**	**193%**	**0.28s**	**93.1%**
		Bosh3	0.048s	0.06s	25%	0.048s	14.5%	0.05s	4.1%	**0.085s**	**88%**	**0.07s**	**55.5%**
	MNIST	Dopri5	0.04s	0.042s	5%	0.0401s	1%	0.041s	2.5%	**0.07s**	**75%**	**0.0625s**	**56.25%**
		AdapHeun	0.1s	0.11s	10%	0.11s	10%	0.11s	10%	**0.435s**	**335%**	**0.15s**	**56.4%**
		Bosh3	0.036s	0.036s	0%	0.036s	0%	0.035	-2.7%	**0.09s**	**150%**	**0.62s**	**72.2%**

Table 2: ITP and ETP values for measuring transferability between Dopri5,Bosh3 and Adaptive Heun solvers. *TS* represents Target Solver and *BS* represents Base Solver.

		CIFAR-10			MNIST		
Type	TS / BS	Dopri5	AH	Bosh3	Dopri5	AH	Bosh3
ITP	Dopri5	–	70.0	80	–	100	90
	AH	92.5	–	70.0	22	–	100
	Bosh3	20	80	–	40	100	–
ETP	Dopri5	–	9.1	1.1	–	22.7	2.2
	AH	26.3	–	22.2	3.9	–	5.5
	Bosh3	0.1	1.5	–	7.5	2.5	–

images randomly. To evaluate solver-based transferability, we consider two aforementioned solver with same network architecture. Table 2 shows our findings. We can observe that transferability can exist between two solvers because of consistently high ITP and ETP values. It can be noticed that for CIFAR-10 dataset, created adversarial inputs for Adaptive Heun are more effective against Dopri5 solver and Bosh3 solver in terms of transferability. However, for MNIST data, created adversarial inputs for Dopri5 are more transferable. For Bosh3 solver, transferability to the Adaptive Heun solver is significantly higher than transferability to the Dopri5. To evaluate network-based transferability, we consider Adaptive Heun ODE solver with two different Neural ODE architectures.

To evaluate network-based transferability, we consider Adaptive Heun ODE solver with two different Neural ODE architecture (M1 and M2). M1 is the larger model that has an extra convolutional layer than M2. Table 3 shows the results. It can be noticed that for MNIST dataset, ETP values are significantly higher than ETP values for CIFAR-10 dataset. However, ITP values are high for all the scenarios. Therefore, we can observe that cross-architectural transferability is feasible.

Table 3: ITP and ETP values for measuring transferability between Larger model (M1) and Smaller Model (M2). *TM* represents Target Model and *BM* represents Base Model.

		CIFAR-10		MNIST	
Type	TM / BM	M1	M2	M1	M2
ITP	M1	–	99	–	99
	M2	99	–	99	–
ETP	M1	–	7	–	33.4
	M2	7.9	–	20.6	–

. Ablation Study

We perform an ablation study to show the impact of using importance scores. For that purpose, instead of optimizing the step-sizes based on the importance, we put same weightage for all step-sizes. We find that effectiveness of restricted attack can decrease if we do not include the importance of step-sizes. For MNIST data, the average latency caused by the restricted attack inputs is decreased by 6.3% for Dopri5 solver, whereas, for Adaptive-Heun solver, the average latency caused by the restricted attack inputs is decreased by 2.2%. Although for Bosh3 solver, the impact of importance score is higher because the average latency caused by the restricted attack inputs is decreased by 25.8%. For CIFAR-10 data, the effect of importance score is higher for Dopri5 and Adaptive Heun solvers as the average latency caused by the restricted attack can be decreased by 13.1% and 9.2% for Dopri5 and Adaptive-Heun solver respectively. For Bosh3 solver, the average latency caused by the restricted attack can be decreased by 21.4% without using the importance score. the Though, for unrestricted attack, the induced latency does not decrease significantly if we don't include importance Scores during optimization.

6. Discussion

In this section, we discuss the following topics: Accuracy Robustness vs Efficiency Robustness, Effect of Corruption on the Seed Input latency correlation between ODE solvers.

. Accuracy Robustness vs Efficiency Robustness

In this section, we discuss the relation between accuracy robustness and efficiency robustness. First, we randomly select 1000 CIFAR-10 and MNIST seed images and generate adversarial images with restricted attack for Dopri-5 solver. We found that the output labels for 90.2% of the CIFAR-10 test inputs are the same as benign ones. However, for MNIST data, only 11% output labels of the test inputs are the same as benign inputs. Thus, while we can not conclude that the accuracy robustness and efficiency robustness of Neural ODEs are related, we can notice that the Neural ODE model trained with the CIFAR-10 dataset is more robust than the ODE model trained with the MNIST dataset.

. Effect of Corruption on the Seed Input

In this section, we discuss the effect of corruption on the seed input on the restricted attack method. Our objective is to understand if an input is already corrupted, how much latency can be increased through AntiNODE. For this purpose, we select three types of common corruptions: Fog, Frost, and Snow. These corruptions are applied on CIFAR-10 inputs, and these inputs are used as seed inputs to restricted attack method. We calculate the mean percentage latency increase for each scenario. We use Dopri-5 solver and Adaptive Heun solvers for this experiment.

Table 4: Percentage latency increase for different corruptions in seed inputs

Classifier / Solver	Fog	Frost	Snow
Dopri5	25.6%	25.3%	25.6%
Adaptive Heun	61.8%	52.8%	52.6%

Table 4 shows the results. We observe that the mean percentage latency increase for Dopri5 solver similar for all the corruptions, and also the mean percentage latency increases for corrupted seeds is are lesser than mean percentage latency increases for ordinary seeds. However, for Adaptive Heun solver, the mean percentage latency increases for corrupted seeds become significantly higher than mean percentage latency increases for ordinary seeds. Therefore, the effect of corruption on seed input can vary based on solvers.

. Latency correlation between ODE solvers

In this section, we try to explore the correlation between percentage of latency increased between two solvers for inputs modified by restricted attack. The objective of the section is to find out if inputs with a high latency increase for one solver would have a high latency increase for the other solver also. We use Pearson Correlation [1] for this purpose, which is one of the metrics that can find the strength of the relationship between two variables. For MNIST data, we find that the Pearson Correlation Coeff (r) and p-value between the percentage of latency increased for both solvers are -0.08 and 0.58, respectively. Similarly, for CIFAR-10 data, the Pearson Correlation Coeff (r) and p-value for both solvers are 0.13 and 0.02. We can conclude from the results that for CIFAR-10 data, the correlation between the percentage of latency increased for both solvers exists because of low p-value.

7. Conclusion

In this paper, we have proposed AntiNODE [1] to show the vulnerability of Neural ODEs against latency-surging test inputs. Here, we have proposed two types of attack methods. To the best of our knowledge, we are the first to explore latency-based vulnerabilities in Neural ODEs. We also observe that adversarial examples generated by AntiNODE can be transferable. Finally, we show that test inputs can improve the efficiency robustness of Neural ODE models.

[1]https://github.com/anonymous2015258/NODEAttack

References

[1] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 37–40. Springer, 2009.

[2] Przemyslaw Bogacki and Lawrence F Shampine. A 3 (2) pair of runge-kutta formulas. *Applied Mathematics Letters*, 2(4):321–325, 1989.

[3] Aleksandar Bojchevski and Stephan Günnemann. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *Proceedings of the International Conference on Machine Learning*, pages 695–704, 2019.

[4] T. B. Brown, N. Carlini, C. Zhang, C. Olsson, P. Christiano, and I. Goodfellow. Unrestricted adversarial examples. *arXiv preprint arXiv:1809.08352*, 2018.

[5] Arif Budiman, Mohamad Ivan Fanany, and Chan Basaruddin. Distributed averaging cnn-elm for big data. *arXiv preprint arXiv:1610.02373*, 2016.

[6] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden Voice Commands. In *Proceedings of the USENIX Security Symposium*, pages 513–530, 2016.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[8] Fabio Carrara, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. On the robustness to adversarial examples of neural ode image classifiers. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

[9] Dr Philippe Cattin. Image restoration: Introduction to signal and image processing. *MIAC, University of Basel. Retrieved*, 11:93, 2013.

[10] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018.

[11] Simin Chen, Hanlin Chen, Mirazul Haque, Cong Liu, and Wei Yang. The dark side of dynamic routing neural networks: Towards efficiency backdoor injection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24585–24594, 2023.

[12] Simin Chen, Mirazul Haque, Cong Liu, and Wei Yang. Deepperform: An efficient approach for performance testing of resource-constrained neural networks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

[13] Simin Chen, Cong Liu, Mirazul Haque, Zihe Song, and Wei Yang. Nmtsloth: understanding and testing efficiency degradation of neural machine translation systems. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1148–1160, 2022.

[14] Simin Chen, Zihe Song, Mirazul Haque, Cong Liu, and Wei Yang. Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15365–15374, 2022.

[15] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial Classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 99–108, 2004.

[16] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[17] D.Lowd and C.Meek. Good Word Attacks on Statistical Spam Filters. In *Proceedings of the Second Conference on Email and Anti-Spam*, 2005.

[18] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

[19] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *arXiv preprint arXiv:1904.01681*, 2019.

[20] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018.

[21] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[22] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, 2014.

[23] Alex Graves. Adaptive Computation Time for Recurrent Neural Networks. *arXiv preprint arXiv:1603.08983*, 2016.

[24] Mirazul Haque, Christof J Budnik, and Wei Yang. Corrgan: Input transformation technique against natural corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 194–197, 2022.

[25] Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. ILFO: Adversarial Attack on Adaptive Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14264–14273, 2020.

[26] Mirazul Haque, Simin Chen, Wasif Arman Haque, Cong Liu, and Wei Yang. Nodeattack: Adversarial attack on the energy consumption of neural odes. 2021.

[27] Mirazul Haque, Rutvij Shah, Simin Chen, Berrak Şişman, Cong Liu, and Wei Yang. Slothspeech: Denial-of-service attack against speech recognition models. *arXiv preprint arXiv:2306.00794*, 2023.

[28] Mirazul Haque, Yaswanth Yadlapalli, Wei Yang, and Cong Liu. Ereba: Black-box energy testing of adaptive neural networks. *arXiv preprint arXiv:2202.06084*, 2022.

[29] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[30] Sanghyun Hong, Yiğitcan Kaya, Ionuţ-Vlad Modoranu, and Tudor Dumitraş. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. *arXiv preprint arXiv:2010.02432*, 2020.

[31] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. In *Advances in Neural Information Processing Systems*, pages 1884–1894, 2019.

[32] S Jayaraman, S Esakkirajan, and T Veerakumar. Digital image processing tmh publication. *Year of Publication*, pages 444–542, 2009.

[33] Robin Jia and Percy Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. *arXiv preprint arXiv:1707.07328*, 2017.

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] Wilhelm Kutta. Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.*, 46:435–453, 1901.

[36] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[37] Daniel Lowd and Christopher Meek. Adversarial Learning. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 641–647, 2005.

[38] James Lu, Kaiwen Deng, Xinyuan Zhang, Gengbo Liu, and Yuanfang Guan. Neural-ode for pharmacokinetics modeling and its advantage to alternative machine learning models in predicting new dosing regimens. *Iscience*, 24(7):102804, 2021.

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[40] Zoya Meleshkova, Sergei Evgenievich Ivanov, and Lubov Ivanova. Application of neural ode with embedded hybrid method for robotic manipulator control. *Procedia Computer Science*, 193:314–324, 2021.

[41] James Morrill, Patrick Kidger, Lingyi Yang, and Terry Lyons. Neural controlled differential equations for online prediction tasks. *arXiv preprint arXiv:2106.11028*, 2021.

[42] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019.

[43] Sunghyun Park, Kangyeol Kim, Junsoo Lee, Jaegul Choo, Joonseok Lee, Sookyung Kim, and Edward Choi. Vid-ode: Continuous-time video generation with neural ordinary differential equation. *arXiv preprint arXiv:2010.08188*, 2020.

[44] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

[45] Alessio Quaglino, Marco Gallieri, Jonathan Masci, and Jan Koutnık. Accelerating neural odes with spectral elements. *arXiv preprint arXiv:1906.07038*, 2019.

[46] Carl Runge. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

[47] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *Advances in Neural Information Processing Systems*, 31, 2018.

[48] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[49] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199*, 2013.

[50] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast Inference via Early Exiting from Deep Neural Networks. In *Proceedings of the International Conference on Pattern Recognition*, pages 2464–2469, 2016.

[51] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning Dynamic Routing in Convolutional Networks. In *Proceedings of the European Conference on Computer Vision*, pages 409–424, 2018.

[52] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[53] Hanshu Yan, Jiawei Du, Vincent YF Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. *arXiv preprint arXiv:1910.05513*, 2019.

[54] Guanqun Yang, Mirazul Haque, Qiaochu Song, Wei Yang, and Xueqing Liu. Testaug: A framework for augmenting capability-based nlp tests. *arXiv preprint arXiv:2210.08097*, 2022.

[55] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[56] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial Attacks on Neural Networks for Graph Data. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 2847–2856, 2018.