Neural ODE

Background on ODE

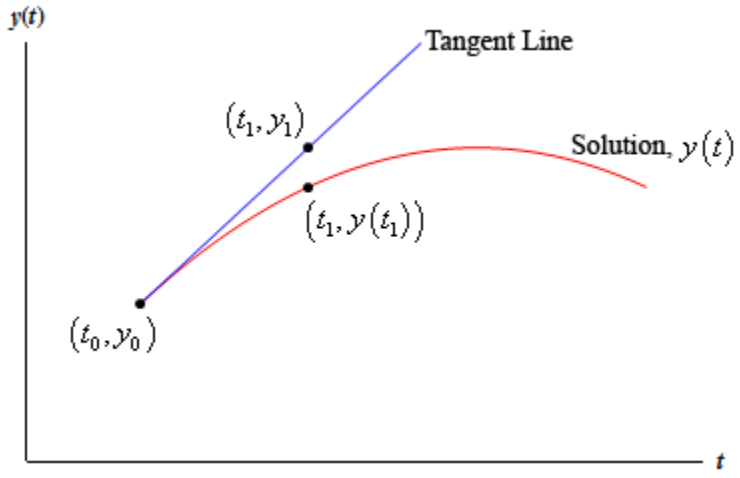
1. General equation:

$$\frac{dy}{dt} = f(t, y) \qquad y(t_0) = y_0$$

2. Discretized Solution:

$$y_{n+1} = y_n + f(t_n, y_n) \cdot (t_{n+1} - t_n)$$

This comes from Euler's method:



ResNet

It can be seen as an ODE solution:

Models such as residual networks, recurrent neural network decoders, and normalizing flows build complicated transformations by composing a sequence of transformations to a hidden state:

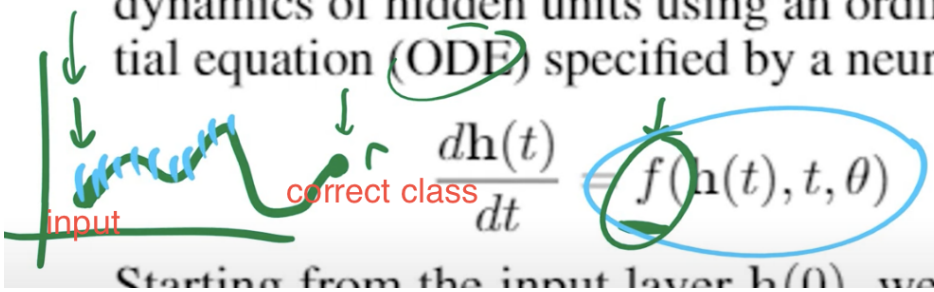
$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \quad (1)$$

where $t \in \{0 \dots T\}$ and $\mathbf{h}_t \in \mathbb{R}^D$. These iterative updates can be seen as an Euler discretization of a continuous transformation (Lu et al., 2017; Haber and Ruthotto, 2017; Ruthotto and Haber, 2018).

Note that in equation (1), theta depends on time. (We will get back on this in a while)

NODE Idea

What happens as we add more layers and take smaller steps? In the limit, we parameterize the continuous dynamics of hidden units using an ordinary differential equation (ODE) specified by a neural network:



$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta) \quad (2)$$

Starting from the input layer $\mathbf{h}(0)$, we can define the

Note that in equation (2), theta is the parameters of f, and it does not depend on time. (We will get back on this in a while)

Hence, instead of taking advantage of discrete Euler Solver, which accumulates error in each step, let's use an blackbox ODE Solver, which is a function provides the solution of the Ordinary Differential equation.

Note

From a rigorous point of view, NODE is not actually a continuous version of ResNet. Actually (1) formula is a special type of ResNet whose weights are identical at every layer. This

(identical f at different layers) limits the expressive power of NODE and merely learns simple processes. There are some works based on NODE to model richer families of functions.

Backpropagation in NODE (How to learn theta)

$$z(1) = z(0) + \int_0^1 f(z(t), \theta) dt$$

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta) dt\right) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right)$$

1. First Idea: Backpropagate through the operations of ODESolve -> Not a good idea:
    a. Memory Cost (The computation graph can get extremely large in the solver)
    b. Do not differentiate the approximation.
2. Adjoint Sensitivity:
    a. First define how loss depends on hidden states, which is called adjoint:
    $$\mathbf{a}(t) = \partial L / \partial \mathbf{z}(t).$$
    b. Then one can show it itself can be formulated by another ODE:
    $$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\mathsf{T} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$
    c. An ODE Solver can run backwards, where its initial state is ∂L/∂z(t1).
        i. Key Observation: We do not have to to save z(t) in forward pass as we can recompute z(t) backwards together with a(t) by using z(t1)
    d. One can show that the gradients w.r.t to parameters are as follows:
    $$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^\mathsf{T} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \theta} dt$$
    e. The integrals can be computed by a call to ODE Solver.
    f. An interesting proof of this process by lagrange multipliers

Notion of Depth in NODE

They do not have a fixed number of layers. Best analogy to depth is the number of evaluations of the dynamics network of the ODE Solver makes. It is like a query to the function f

Is Neural ODE a misleading useless paper?[1]

Robustness of NODE

1) On Robustness of Neural Ordinary Differential Equations[2]

a) Introduction

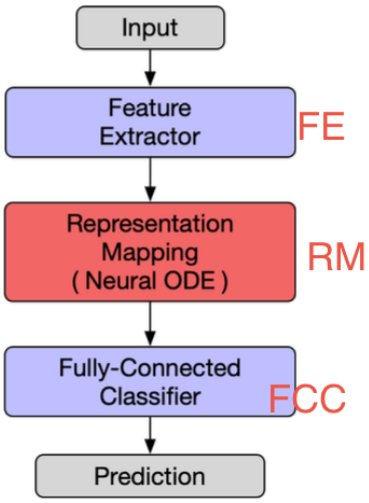
ABSTRACT

Neural ordinary differential equations (ODEs) have been attracting increasing attention in various research domains recently. There have been some works studying optimization issues and approximation capabilities of neural ODEs, but their robustness is still yet unclear. In this work, we fill this important gap by exploring robustness properties of neural ODEs both empirically and theoretically. We first present an empirical study on the robustness of the neural ODE-based networks (ODENets) by exposing them to inputs with various types of perturbations and subsequently investigating the changes of the corresponding outputs. In contrast to conventional convolutional neural networks (CNNs), we find that the ODENets are more robust against both random Gaussian perturbations and adversarial attack examples. We then provide an insightful understanding of this phenomenon by exploiting a certain desirable property of the flow of a continuous-time ODE, namely that integral curves are non-intersecting. Our work suggests that, due to their intrinsic robustness, it is promising to use neural ODEs as a basic block for building robust deep network models. To further enhance the robustness of vanilla neural ODEs, we propose the time-invariant steady neural ODE (TisODE), which regularizes the flow on perturbed data via the time-invariant property and the imposition of a steady-state constraint. We show that the TisODE method outperforms vanilla neural ODEs and also can work in conjunction with other state-of-the-art architectural methods to build more robust deep networks.

[1] "David Duvenaud | Reflecting on Neural ODEs ... - YouTube." 15 Dec. 2019, https://www.youtube.com/watch?v=YZ-_E7A3V2w. Accessed 10 Jan. 2021.
[2] "On Robustness of Neural Ordinary Differential Equations." 12 Oct. 2019, https://arxiv.org/abs/1910.05513. Accessed 4 Jan. 2021.

b) Architecture:

Architectures: On the MNIST dataset, both the ODENet and the CNN model consists of four convolutional layers and one fully-connected layer. The total number of parameters of the two models is around 140k. On the SVHN dataset, the networks are similar to those for the MNIST; we only changed the input channels of the first convolutional layer to three. On the ImgNet10 dataset, there are nine convolutional layers and one fully-connected layer for both the ODENet and the CNN model. The numbers of parameters is approximately 280k. In practice, the neural ODE can be solved with different numerical solvers such as the Euler method and the Runge-Kutta methods (Chen et al., 2018). Here, we use the easily-implemented Euler method in the experiments. To balance the computation and the continuity of the flow, we solve the ODE initial value problem in equation (1) by the Euler method with step size 0.1. Our implementation builds on the open-source neural ODE codes.[1] Details on the network architectures are included in the Appendix.



For a fair comparison with conventional CNN models, we made sure that the number of parameters of an ODENet is close to that of its counterpart CNN model. Specifically, the ODENet shares the same network architecture with the CNN model for the FE and FCC parts. The only difference is that, for the RM part, the input of the ODE-based RM is concatenated with one more channel which represents the time t, while the RM in a CNN model has a skip connection and serves as a residual block. During the training phase, all the hyperparameters are kept the same, including training epochs, learning rate schedules, and weight decay coefficients. Each model is trained *three* times with different random seeds, and we report the average performance (classification accuracy) together with the standard deviation.

c) Some doubts about the choice of designs:

i) It is stated in the paper they used common backpropagation computation graph. Why not the proposed adjoint sensitivity method?

ii) It is stated in the paper that they used Euler method as ODE Solver. Isn't it equal to a very deep ResNet with equal function in each layer in a way?

As far as I understood, we wanted to take advantage of modern ODE Solvers. How is the robustness with those solvers?

d) Empirical Results:

i) Using original datasets:

	Gaussian noise			Adversarial attack		
MNIST	$\sigma = 50$	$\sigma = 75$	$\sigma = 100$	FGSM-0.15	FGSM-0.3	FGSM-0.5
CNN	98.1±0.7	85.8±4.3	56.4±5.6	63.4±2.3	24.0±8.9	8.3±3.2
ODENet	**98.7**±0.6	**90.6**±5.4	**73.2**±8.6	**83.5**±0.9	**42.1**±2.4	**14.3**±2.1
SVHN	$\sigma = 15$	$\sigma = 25$	$\sigma = 35$	FGSM-3/255	FGSM-5/255	FGSM-8/255
CNN	90.0±1.2	76.3±2.7	60.9±3.9	29.2±2.9	13.7±1.9	5.4±1.5
ODENet	**95.7**±0.7	**88.1**±1.5	**78.2**±2.1	**58.2**±2.3	**43.0**±1.3	**30.9**±1.4
ImgNet10	$\sigma = 10$	$\sigma = 15$	$\sigma = 25$	FGSM-5/255	FGSM-8/255	FGSM-16/255
CNN	80.1±1.8	63.3±2.0	40.8±2.7	28.5±0.5	18.1±0.7	9.4±1.2
ODENet	**81.9**±2.0	**67.5**±2.0	**48.7**±2.6	**36.2**±1.0	**27.2**±1.1	**14.4**±1.7

ii) Datasets augmented with Gaussian Perturbations:

	Gaussian noise	Adversarial attack			
MNIST	$\sigma = 100$	FGSM-0.3	FGSM-0.5	PGD-0.2	PGD-0.3
CNN	98.7±0.1	54.2±1.1	15.8±1.3	32.9±3.7	0.0±0.0
ODENet	**99.4**±0.1	**71.5**±1.1	**19.9**±1.2	**64.7**±1.8	**13.0**±0.2
SVHN	$\sigma = 35$	FGSM-5/255	FGSM-8/255	PGD-3/255	PGD-5/255
CNN	90.6±0.2	25.3±0.6	12.3±0.7	32.4±0.4	14.0±0.5
ODENet	**95.1**±0.1	**49.4**±1.0	**34.7**±0.5	**50.9**±1.3	**27.2**±1.4
ImgNet10	$\sigma = 25$	FGSM-5/255	FGSM-8/255	PGD-3/255	PGD-5/255
CNN	92.6±0.6	40.9±1.8	26.7±1.7	28.6±1.5	11.2±1.2
ODENet	92.6±0.5	**42.0**±0.4	**29.0**±1.0	**29.8**±0.4	**12.3**±0.6

e) Intuition about the robustness of NODE:

i) The goal is to show that a small change on the feature map extracted by FE (because of perturbation in input) will not lead to a large deviation in the output of RM. (In order to control the change of feature maps, they regularized weights of FE by weight decay)

ii) The fundamental theorem here is:

> **Theorem 1** (ODE integral curves do not intersect (Coddington & Levinson, 1955; Younes, 2010; Dupont et al., 2019)). *Let $\mathbf{z}_1(t)$ and $\mathbf{z}_2(t)$ be two solutions of the ODE in (1) with different initial conditions, i.e. $\mathbf{z}_1(0) \neq \mathbf{z}_2(0)$. In (1), f_θ is continuous in t and globally Lipschitz continuous in $\mathbf{z}$. Then, it holds that $\mathbf{z}_1(t) \neq \mathbf{z}_2(t)$ for all $t \in [0, \infty)$.*

So it means the solutions will not even intersect with each other. As an 1-d example:

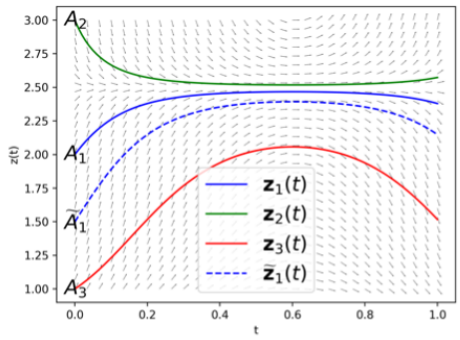


Figure 2: No integral curves intersect. The integral curve starting from $\widetilde{A_1}$ is always sandwiched between two integral curves starting from A_1 and A_3.

z_1(t), z_2(t), z_3(t) are three solutions for an ODE with three different initial state A_1 = (0, z_1(0)), A_2 = (0, z_2(0)), A_3 = (0, z_3(0)), where z_i(0) can depict the feature map of i_th datum. By the theorem, if A_1 is between A_2 and A_3, z_1(t) is always sandwiched between z_2(t) and z_3(t).

iii) Now in terms of perturbation:

Now, let $\epsilon < \min\{|z_2(0) - z_1(0)|, |z_3(0) - z_1(0)|\}$. Consider a solution $\widetilde{z}_1(t)$ of equation (1). The integral curve $\widetilde{z}_1(t)$ starts from a point $\widetilde{A}_1 = (0, \widetilde{z}_1(0))$. The point $\widetilde{A}_1$ is in the ϵ-neighborhood of A_1 with $|\widetilde{z}_1(0) - z_1(0)| < \epsilon$. By Theorem 1, we know that $|\widetilde{z}_1(T) - z_1(T)| \leq |z_3(T) - z_2(T)|$. In other words, if any perturbation smaller than ϵ is added to the scalar $z_1(0)$ in A_1, the deviation from the original output $z_1(T)$ is bounded by the distance between $z_2(T)$ and $z_3(T)$. In contrast, in a CNN model, there is no such bound on the deviation from the original output. Thus, we opine that due to this non-intersecting property, ODENets are intrinsically robust.

iv) Doubts:

(1) What are z_2 and z_3? Are they feature maps from other classes? Do they represent decision boundaries?

(2) How much large can epsilon get?

f) TisODE

i) First they show with current formulation of Neural ODE, one cannot find a useful upper bound.

Theorem 2. *Let $U \subset \mathbb{R}^d$ be an open set. Let $f : U \times [0, T] \to \mathbb{R}^d$ be a continuous function and let $\mathbf{z}_1$, $\mathbf{z}_2$: $[0, T] \to U$ satisfy the initial value problems:*

$$\frac{\mathrm{d}\mathbf{z}_1(t)}{\mathrm{d}t} = f(\mathbf{z}_1(t), t), \quad \mathbf{z}_1(t) = \mathbf{x}_1$$
$$\frac{\mathrm{d}\mathbf{z}_2(t)}{\mathrm{d}t} = f(\mathbf{z}_2(t), t), \quad \mathbf{z}_2(t) = \mathbf{x}_2$$

Assume there is a constant $C \geq 0$ such that, for all $t \in [0, T]$,

$$\|f(\mathbf{z}_2(t), t) - f(\mathbf{z}_1(t), t))\| \leq C\|\mathbf{z}_2(t) - \mathbf{z}_1(t)\|$$

Then, for any $t \in [0, T]$,

$$\|\mathbf{z}_1(t) - \mathbf{z}_2(t)\| \leq \|\mathbf{x}_2 - \mathbf{x}_1\| \cdot e^{Ct}.$$

ii) TisODE Goal: Assume the l_p ball of the feature map of a datum $z_1(0)$ (they are considered as the perturbed examples). Its corresponding curve is $z_1(t)$, The goal is to show at time T (the output of NODE), there is an upper bound for the difference of $z_1(T)$ and the output of the perturbed example called $z'_1(T)$. The interesting fact is that the upper bound is only dependent on $z_1(t)$.

iii) To reach this goal, considering (i), they proposed to make the ODE time-invariant:

$$\begin{cases} \dfrac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = f_\theta(\mathbf{z}(t)); \\ \mathbf{z}(0) = \mathbf{z}_{\text{in}}; \\ \mathbf{z}_{\text{out}} = \mathbf{z}(T). \end{cases} \tag{2}$$

iv) Then we have:

Let $\mathbf{z}_1(t)$ be a solution of (2) on $[0, \infty)$ and $\epsilon > 0$ be a small positive value. We define the set $\mathbb{M}_1 = \{(\mathbf{z}_1(t), t) | t \in [0, T], \|\mathbf{z}_1(t) - \mathbf{z}_1(0)\| \leq \epsilon\}$. This set contains all points on the curve of $\mathbf{z}_1(t)$ during $[0, T]$ that are also inside the ϵ-neighborhood of $\mathbf{z}_1(0)$. For some element $(\mathbf{z}_1(T'), T') \in \mathbb{M}_1$, let $\widetilde{\mathbf{z}}_1(t)$ be the solution of (2) which starts from $\widetilde{\mathbf{z}}_1(0) = \mathbf{z}_1(T')$. Then we have

$$\widetilde{\mathbf{z}}_1(t) = \mathbf{z}_1(t + T') \tag{3}$$

for all t in $[0, \infty)$. The property shown in equation (3) is known as the *time-invariant property*. It indicates that the integral curve $\widetilde{\mathbf{z}}_1(t)$ is the $-T'$ shift of $\mathbf{z}_1(t)$ (Figure 3).

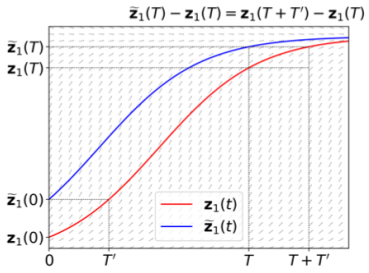


Figure 3: An illustration of the time-invariant property of ODEs. We can see that the curve $\widetilde{\mathbf{z}}_1(t)$ is exactly the horizontal translation of $\mathbf{z}_1(t)$ on the interval $[T', \infty)$.

v) Now we can find the upper bound:

We can regard $\widetilde{\mathbf{z}}_1(0)$ as a slightly perturbed version of $\mathbf{z}_1(0)$, and we are interested in how large the difference between $\widetilde{\mathbf{z}}_1(T)$ and $\mathbf{z}_1(T)$ is. In a robust model, the difference should be small. By equation (3), we have $\|\widetilde{\mathbf{z}}_1(T) - \mathbf{z}_1(T)\| = \|\mathbf{z}_1(T + T') - \mathbf{z}_1(T)\|$. Since $T' \in [0, T]$, the difference between $\mathbf{z}_1(T)$ and $\widetilde{\mathbf{z}}_1(T)$ can be bounded as follows,

$$\|\widetilde{\mathbf{z}}_1(T) - \mathbf{z}_1(T)\| = \left\| \int_T^{T+T'} f_\theta(\mathbf{z}_1(t))\, \mathrm{d}t \right\| \leq \left\| \int_T^{T+T'} |f_\theta(\mathbf{z}_1(t))|\, \mathrm{d}t \right\| \leq \left\| \int_T^{2T} |f_\theta(\mathbf{z}_1(t))|\, \mathrm{d}t \right\|, \tag{4}$$

where all norms are ℓ_2 norms and $|f_\theta|$ denotes the element-wise absolute operation of a vector-valued function f_θ. That is to say, the difference between $\widetilde{\mathbf{z}}_1(T)$ and $\mathbf{z}_1(T)$ can be bounded by only using the information of the curve $\mathbf{z}_1(t)$. For any $t' \in [0, T]$ and element $(\mathbf{z}_1(t'), t') \in \mathbb{M}_1$, consider the integral curve that starts from $\mathbf{z}_1(t')$. The difference between the output state of this curve and $\mathbf{z}_1(T)$ satisfies inequality (4).

vi) As I mentioned, the upper bound is only dependent on $z_1(t)$. Hence, we can add this term to our loss which assures the output of perturbed examples would not be that different from the clean example.

invariant neural ODE.

$$L_{\text{ss}} = \sum_{i=1}^{N} \left\| \int_T^{2T} |f_\theta(\mathbf{z}_i(t))|\, \mathrm{d}t \right\|, \tag{5}$$

where N is the number of samples in the training set and $\mathbf{z}_i(t)$ is the solution whose initial state equals to the feature of the i^{th} sample. The regularization term L_{ss} is termed as the steady-state loss. This terminology "steady state" is borrowed from the dynamical systems literature. In a stable dynamical system, the states stabilize around a fixed point, known as the *steady-state*, as time tends to infinity. If we can ensure that L_{ss} is small, for each sample, the outputs of all the points in $\mathbb{M}_i$ will stabilize around $\mathbf{z}_i(T)$. Consequently, the model is robust. This modification of the neural ODE is dubbed *Time-invariant steady neural ODE*.

vii) Results of TisODE:

Table 3: Classification accuracy (mean ± std in %) on perturbed images from MNIST, SVHN and ImgNet10. To evaluate the robustness of classifiers, we use three types of perturbations, namely zero-mean Gaussian noise with standard deviation σ, FGSM attack and PGD attack. From the results, the proposed TisODE effectively improve the robustness of the vanilla neural ODE.

	Gaussian noise	Adversarial attack			
MNIST	$\sigma = 100$	FGSM-0.3	FGSM-0.5	PGD-0.2	PGD-0.3
CNN	98.7±0.1	54.2±1.1	15.8±1.3	32.9±3.7	0.0±0.0
ODENet	99.4±0.1	71.5±1.1	19.9±1.2	64.7±1.8	13.0±0.2
TisODE	**99.6**±0.0	**75.7**±1.4	**26.5**±3.8	**67.4**±1.5	**13.2**±1.0
SVHN	$\sigma = 35$	FGSM-5/255	FGSM-8/255	PGD-3/255	PGD-5/255
CNN	90.6±0.2	25.3±0.6	12.3±0.7	32.4±0.4	14.0±0.5
ODENet	**95.1**±0.1	49.4±1.0	34.7±0.5	50.9±1.3	27.2±1.4
TisODE	94.9±0.1	**51.6**±1.2	**38.2**±1.9	**52.0**±0.9	**28.2**±0.3
ImgNet10	$\sigma = 25$	FGSM-5/255	FGSM-8/255	PGD-3/255	PGD-5/255
CNN	92.6±0.6	40.9±1.8	26.7±1.7	28.6±1.5	11.2±1.2
ODENet	92.6±0.5	42.0±0.4	29.0±1.0	29.8±0.4	12.3±0.6
TisODE	**92.8**±0.4	**44.3**±0.7	**31.4**±1.1	**31.1**±1.2	**14.5**±1.1

viii) The paper proposes it can be used in conjunction with other techniques. But as far as I know, at least one of the techniques they mention is known as obfuscated gradient[3]. SO WHAT IS THE POINT??

ix) We already know that NODE has a limited expressive power. Making it time-invariant does not weaken its power again?

g) Results in the setting of Adversarial Training:

	Gaussian noise	Adversarial attack		
MNIST	$\sigma = 100$	FGSM-0.3	FGSM-0.5	PGD-0.3
CNN	58.0	98.4	21.1	5.3
ODENet	84.2	99.1	36.0	12.3
TisODE	**87.9**	**99.1**	**66.5**	**78.9**

Note that In response to the limitations of adversarial training, designing network architecture towards natural training as robust as adversarial training has received significant attention in recent years, that is why they proposed TisODE even though it is much weaker than the state-of-the-art result by adversarial training.

h) Drawbacks:

i) Still much weaker than adversarial training.

i) Review of Comments from OpenReview[4]:

i) "Since the paper already involves the evaluation using adversarial examples, it will make the paper much more stronger to show that when training both the new approach and the CNN with adversarial training, the proposed regularization can still lead to better robustness" (why?)

[3] "Obfuscated Gradients Give a False Sense of Security ...." https://arxiv.org/abs/1802.00420. Accessed 10 Jan. 2021.

[4] "On Robustness of Neural Ordinary Differential Equations ...." 16 Oct. 2019, https://openreview.net/forum?id=B1e9Y2NYvS. Accessed 10 Jan. 2021.

(1) OR Answer: We can see that the neural ODE-based models are consistently more robust than CNN models. Besides, the proposed TisODE also outperforms the vanilla neural ODE. We have added these experiments to the revision (Appendix section 7.4).

ii) "For neural ODEs, engineers can simply replace a res-block with a neural ODE block, which is also mentioned by David Duvenaud in his NeurIPS talk. Then, training the neural ODE-based networks with our proposed method can yield a robust model."

(1) My Answer: Isn't this wrong? By augmented Neural ODE paper we know that there are functions that NODE cannot express. They even mentioned the problem and the augmented NODE in their paper!

iii) "Another possible improvement of the paper could be to expand the adversarial attacks considered to the gradient-free optimization techniques employed in e.g. [3] which have sharply reduced other defenses against adversarial attacks." (why?)

(1) SPSA attack choosing n=50, T=10, and epsilon = 0.4 : (CNN: avg 33.4%; ODENe: avg 43.1%; TisODE: avg 45.3%)

(2) ZOO [6] with epsilon=0.4: (CNN: avg 15.6%; ODENet: avg 51.0%; TisODE: avg 52.5%)

iv) "Since neural ODE can be considered as a generalization of ResNet to continuous layer depth, isn't it a bit "unfair" to compare neural ODE with CNN instead of ResNet?"

(1) My Answer: They have added this to the paper which is existed in part b (architecture)

(2) My Answer: Another thing they mentioned in the last section in Appendix is that the ResNet layers share the same weight. What would be the results with an actual ResNet?

Table 9: The architecture of the ODENet on CIFAR10.

	Repetition	Layer
FE	×1	Conv(3, 16, 3, 1) + GroupNorm + ReLU
	×1	Conv(16, 32, 3, 2) + GroupNorm + ReLU
	×1	Conv(32, 64, 3, 2) + GroupNorm + ReLU
RM	×2	Conv(64, 64, 3, 1) + GroupNorm + ReLU
FCC	×1	AdaptiveAvgPool2d + Linear(64,10)

7.4.3 AN EXTENSION ON THE COMPARISON BETWEEN CNNS AND ODENETS

Here, we compare CNN and neural ODE-based models by controlling both the number of parameters and the number of function evaluations. We conduct experiments on the MNIST dataset, and all the models are trained only with original non-perturbed images.

For the neural ODE-based models, the time range is set from 0 to 1. We use the Euler method, and the step size is set to be 0.05. Thus the number of evaluations is $1/0.05 = 20$. For the CNN models (specifically ResNet), we repeatedly concatenate the residual block for 20 times, and these 20 blocks share the same weights. Our experiments show that, in this condition, the neural ODE-based models still outperform the CNN models (FGSM-0.15: 87.5% vs. 81.9%, FGSM-0.3: 53.4% vs. 49.7%, PGD-0.2: 11.8% vs. 4.8%).

(3)

v) "The robustness against an adversarial attack means the lack of adversary in the entire adversarial budget, but the analysis in Section 4.1 cannot rule out the perturbation to the points which are not in z1 trajectory but in the adversarial budget."

(1) OR Answer: "In section 4.1, we consider the perturbations that are also on the trajectory of a certain point. A robust model should accurately handle these neighboring points. Thus, the steady-state constraint on these points is a

necessary condition for the robustness. Although this constraint does not include all the neighboring points, it still can contribute to the improvement of robustness."

(2) My Answer: That will happen because of FE right? So what if we remove it?

vi) how many iterations of the PGD attacks do you use in your experiments? According to my own experiments, neural ODE which is trained on unperturbed images might not be very robust to PGD attack with many iterations, like PGD with1,000 iterations.

(1)

References[5][6][7][8][9][10]

[5] "Neural Ordinary Differential Equations." 19 Jun. 2018, https://arxiv.org/abs/1806.07366. Accessed 4 Jan. 2021.

[6] "Neural Ordinary Differential Equations - YouTube." 19 Feb. 2019, https://www.youtube.com/watch?v=jltgNGt8Lpg. Accessed 3 Jan. 2021.

[7] "Neural ODEs: breakdown of another deep learning ...." https://towardsdatascience.com/neural-odes-breakdown-of-another-deep-learning-breakthrough-3e78c7213795. Accessed 3 Jan. 2021.

[8] "Neural Ordinary Differential Equations - Best Paper ... - YouTube." 30 Jan. 2019, https://www.youtube.com/watch?v=V6nGT0Gakyg. Accessed 3 Jan. 2021.

[9] "Neural Ordinary Differential Equations with Envolutionary ...." https://zhouchenlin.github.io/Publications/2019-PRCV-NODE-EW.pdf. Accessed 3 Jan. 2021.

[10] "Deriving the Adjoint Equation for Neural ODEs using ...." 4 Feb. 2020, https://vaipatel.com/deriving-the-adjoint-equation-for-neural-odes-using-lagrange-multipliers/. Accessed 3 Jan. 2021.