Neural ODE

Background on ODE

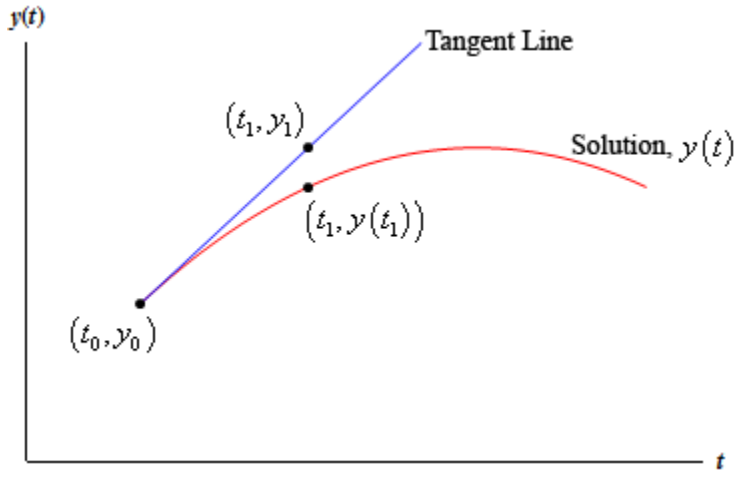
1. General equation:

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

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:

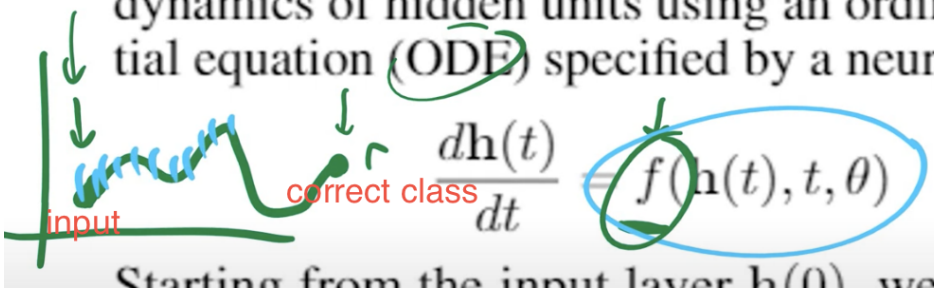
$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t) \quad (1)$$

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

Robustness of NODE

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

a) Introduction

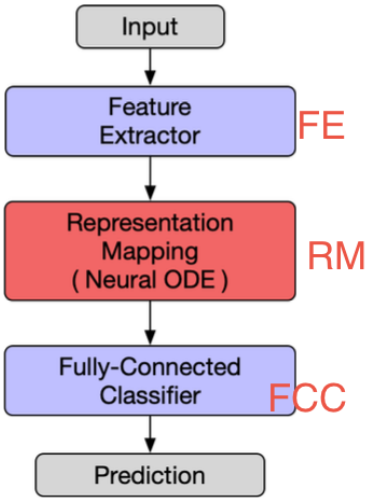
ABSTRACT

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] "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.



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:

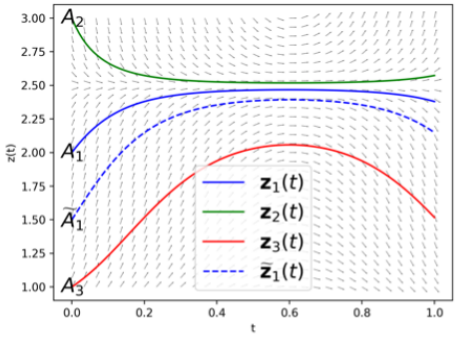


Figure 2: No integral curves intersect. The integral curve starting from $\widetilde{A_1}$ is always sandwiched between two integral curves starting from A_1 and A_3.

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

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.

References[2][3][4][5][6][7]

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

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

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

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

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

[7] "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.