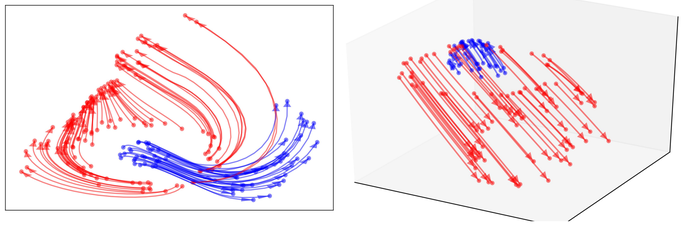
Augmented Neural ODEs

Emilien Dupont, Arnaud Doucet, Yee Whye Teh

emilien.dupont@stats.ox.ac.uk github.com/EmilienDupont/augmented-neural-odes

Summary

- We explore the limitations of Neural ODEs (NODEs) and show that there are functions NODEs cannot represent.
- We introduce Augmented Neural ODEs (ANODEs) which, in addition to being more expressive models, are empirically more stable, generalize better and have a lower computational cost than NODEs.



Learned flows for NODEs (left) and ANODEs (right) mapping input points to linearly separable features for binary classification. ANODEs learn simpler flows that are easier for the ODE solver to compute

Neural ODEs

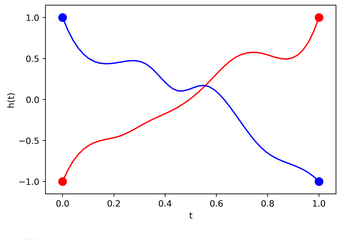
- Neural ODEs map inputs $\mathbf{x}$ into features $\phi(\mathbf{x})$ by solving an ODE:

$$\frac{\mathrm{d}\mathbf{h}(t)}{\mathrm{d}t} = \mathbf{f}(\mathbf{h}(t), t), \qquad \mathbf{h}(0) = \mathbf{x} \tag{1}$$

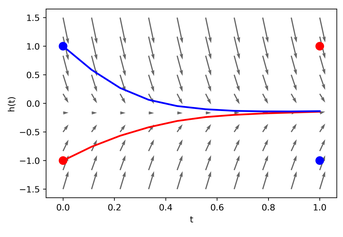
  where $\mathbf{h}(t)$ is the state of the ODE and $\mathbf{f}$ is a learned neural network.
- Features are defined as ODE state at time T, i.e. $\phi(\mathbf{x}) = \mathbf{h}(T)$.
- The function $\mathbf{f}$ is learned through backpropagation.

Example in 1D

- Let $g(x)$ be a function such that $g(1) = -1$ and $g(-1) = 1$.
- Trajectories mapping 1 to -1 and -1 to 1 must intersect (left figure).
- ODE trajectories cannot intersect $\implies$ NODEs cannot represent this function.
- When trained on $g(x)$, NODEs map all points to 0 to minimize mean squared error (right figure).

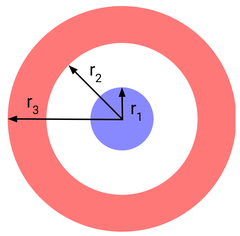


Trajectories mapping 1 to -1 and -1 to 1



Learned vector field with NODE

Functions Neural ODEs cannot represent



- NODEs cannot represent a function that classifies blue region as 1, red region as -1.
- To classify blue as 1, red as -1, need to linearly separate regions. In order to separate them, trajectories must cross, which is not possible. This example generalises to arbitrary number of dimensions.

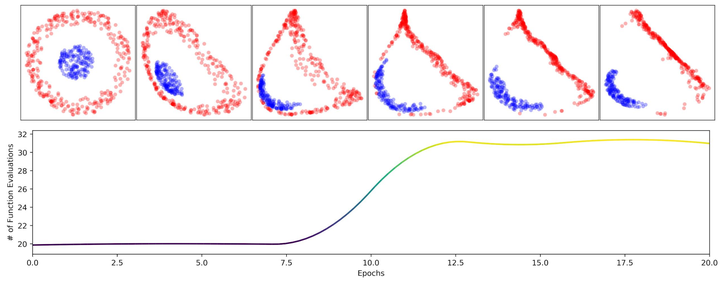
Augmented Neural ODEs

- Solution: append zeros to input to augment the space on which we learn and solve the ODE.
- Allows ODE flow to lift points into additional dimensions to avoid trajectories intersecting each other.

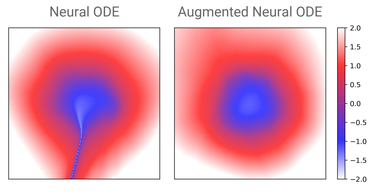
Augmented ODE flow linearly separates points without trajectories crossing

Computational Cost

- During training, learned ODE flows become increasingly complex $\implies$ number of function evaluations (NFEs) required to solve the ODE increases.
- NFEs increase much faster for NODEs than ANODEs, presumably because ANODEs learn simpler flows.
- By learning simpler flows, ANODEs also generalize better.



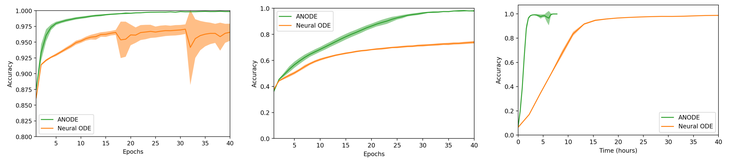
NFEs and feature space for a NODE during training



Plot of how NODEs and ANODEs map points in input space to different outputs

Image datasets

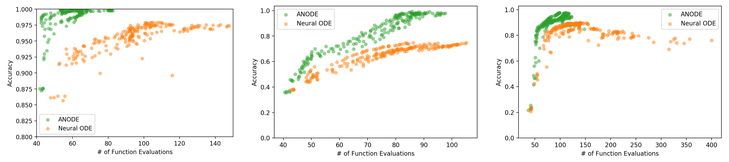
- We compare NODEs with ANODEs on MNIST, CIFAR10, SVHN and 200 classes of 64×64 ImageNet
- For all datasets, ANODEs are 5x faster, have lower computational cost and generalize better



MNIST CIFAR10 ImageNet

Computational Cost

To achieve a given accuracy, ANODEs require fewer NFEs than NODEs $\implies$ ANODEs can model richer classes of functions at a lower computational cost.



MNIST CIFAR10 SVHN

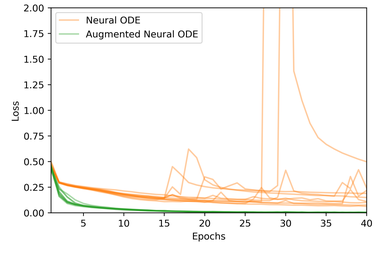
Generalization

ANODEs achieve higher test accuracies and lower losses than NODEs on various datasets.

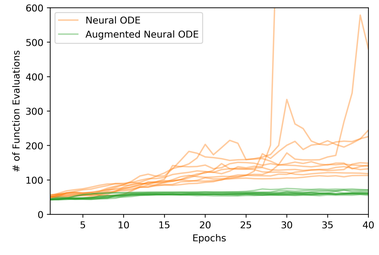
	NODE	ANODE
MNIST	96.4% $\pm$ 0.5	**98.2%** $\pm$ **0.1**
CIFAR10	53.7% $\pm$ 0.2	**60.6%** $\pm$ **0.4**
SVHN	81.0% $\pm$ 0.6	**83.5%** $\pm$ **0.5**

Stability

- NODEs can become prohibitively expensive to train.
- Complex flow leads to unstable training and exploding losses.
- Augmentation consistently leads to stable training and fewer NFEs.



Loss instabilities on MNIST



NFE instabilities on MNIST