Dissecting Neural ODEs

Stefano Massaroli[*,1,3] and Michael Poli[*,2,3], Jinkyoo Park[2], Atsushi Yamashita[1], Hajime Asama[1]

[*]Equal contribution, [1]The University of Tokyo, [2]KAIST, [3]DiffEqML

Generalized Augmented Neural ODEs

Previous result: Neural ODEs cannot represent certain functions (e.g reflections) due to the topology–preserving property of the flows.

Augmented Neural ODEs [Dupont et al. 2019] offer a solution:

$$\begin{bmatrix}\dot{\mathbf{z}}_x(s)\\ \dot{\mathbf{z}}_a(s)\end{bmatrix} = f_{\theta(s)}(s, \mathbf{z}_x, \mathbf{z}_a), \quad \begin{bmatrix}\mathbf{z}_x(0)\\ \mathbf{z}_a(0)\end{bmatrix} = \begin{bmatrix}\mathbf{x}\\ 0\end{bmatrix}$$

- We generalize **augmentation** strategies for Neural ODEs

Input Layer (IL) Augmentation:

$$\begin{bmatrix}\mathbf{z}_x(0)\\ \mathbf{z}_a(0)\end{bmatrix} = h_x(\mathbf{x}), \quad h_x : \mathbb{R}^{n_x} \to \mathbb{R}^{n_z + n_a}$$

- The initial condition of the augmented state is determined by an input network (e.g. a single linear layer)

Higher–Order Neural ODEs

- An alternative **parameter efficient** alternative to augmentation is increasing the order of the differential equation

Higher–Order (HO) Neural ODEs:

$$\begin{cases}\dot{\mathbf{z}}^i(s) = \mathbf{z}^{i+1}(s)\\ \dot{\mathbf{z}}^n(s) = f_{\theta(s)}(s, \mathbf{z}(s))\end{cases} \quad \mathbf{z} = [\mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^n], \; \mathbf{z}^i \in \mathbb{R}^{n_z/n}, \quad f_{\theta(s)} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z/n}$$

e.g. 2nd–order $\begin{bmatrix}\dot{\mathbf{z}}_q(s)\\ \dot{\mathbf{z}}_p(s)\end{bmatrix} = \begin{bmatrix}\mathbf{z}_p\\ f_{\theta(s)}(s, \mathbf{z}_q, \mathbf{z}_p)\end{bmatrix}$

- An alternative **parameter efficient** alternative to augmentation is increasing the order of the differential equation

Ranking Augmentation Strategies

No aug.: expressivity limitations, low performance.
0–aug.: cannot learn the initial condition, **higher NFEs**.
IL aug.: can learn initial conditions, **best performance**.
HO aug: **parameter efficiency**, comparable performance.

A single linear layer is sufficient to relieve vanilla Neural ODEs of their limitations and achieves the best performance.

	NODE		ANODE		IL-NODE		2nd–Ord.	
	MNIST	CIFAR	MNIST	CIFAR	MNIST	CIFAR	MNIST	CIFAR
Test Acc.	96.8	58.9	98.9	70.8	99.1	**73.4**	**99.2**	72.8
NFE	98	93	71	169	44	65	**43**	**59**
Param.[K]	21.4	37.1	20.4	35.0	20.7	36.1	**20.0**	**34.6**

Complete Neural ODE Formulation

The Neural ODE formulation is enhanced

Neural Ordinary Differential Equation

$$\begin{cases}\dot{\mathbf{z}}(s) = f_{\theta(s)}(s, \mathbf{x}, \mathbf{z}(s))\\ \mathbf{z}(0) = h_x(\mathbf{x}) \qquad s \in \mathcal{S}\\ \hat{\mathbf{y}}(s) = h_y(\mathbf{z}(s))\end{cases}$$

Input	$\mathbf{x}$	$\mathbb{R}^{n_x}$
Output	$\hat{\mathbf{y}}$	$\mathbb{R}^{n_y}$
(Hidden) State	$\mathbf{z}$	$\mathbb{R}^{n_z}$
Parameters	$\theta(s)$	$\mathbb{R}^{n_\theta}$
Neural Vector Field	$f_{\theta(s)}$	$\mathbb{R}^{n_z}$
Input Network	h_x	$\mathbb{R}^{n_x} \to \mathbb{R}^{n_z}$
Output Network	h_y	$\mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$

Generalized Adjoint for Neural ODEs

Traditionally, Neural ODEs:

- have constant parameters (i.e. $\theta \in \mathbb{R}^{n_\theta}$)
- are optimized to minimize only terminal loss functions $L(\mathbf{z}(S))$.

$\Longrightarrow$ We consider loss functions

$$\ell = L(\mathbf{z}(S)) + \int_{\mathcal{S}} l(\tau, \mathbf{z}(\tau))\mathrm{d}\tau$$

distributed on the whole depth domain

Generalized Adjoint Gradients

$$\frac{\mathrm{d}\ell}{\mathrm{d}\theta} = \nabla_\theta L + \int_{\mathcal{S}} (\mathbf{a}^\top(\tau)\nabla_\theta f_\theta + \nabla_\theta l)\mathrm{d}\tau \quad \text{where} \quad \begin{matrix}\dot{\mathbf{a}}^\top(s) = -\mathbf{a}^\top(s)\nabla_{\mathbf{z}} f_\theta - \nabla_{\mathbf{z}} l\\ \mathbf{a}^\top(S) = \nabla_{\mathbf{z}(S)} L\end{matrix}$$

Proper Parameter Depth–Variance: $\theta(s)$

- When the model parameters are depth–varying, i.e. $\theta : \mathcal{S} \to \mathbb{R}^{n_\theta}$, we should iterate GD in functional space
- Implementation requires discretizing the problem

Infinite–dim. Adjoint Gradients

Let $\theta(s) \in \mathbb{L}_2(\mathcal{S} \to \mathbb{R}^{n_\theta})$. Then

$$\frac{\delta \ell}{\delta \theta(s)} = \mathbf{a}^\top(s)\frac{\partial f_{\theta(s)}}{\partial \theta(s)}$$

Galerkin and Stacked Neural ODEs

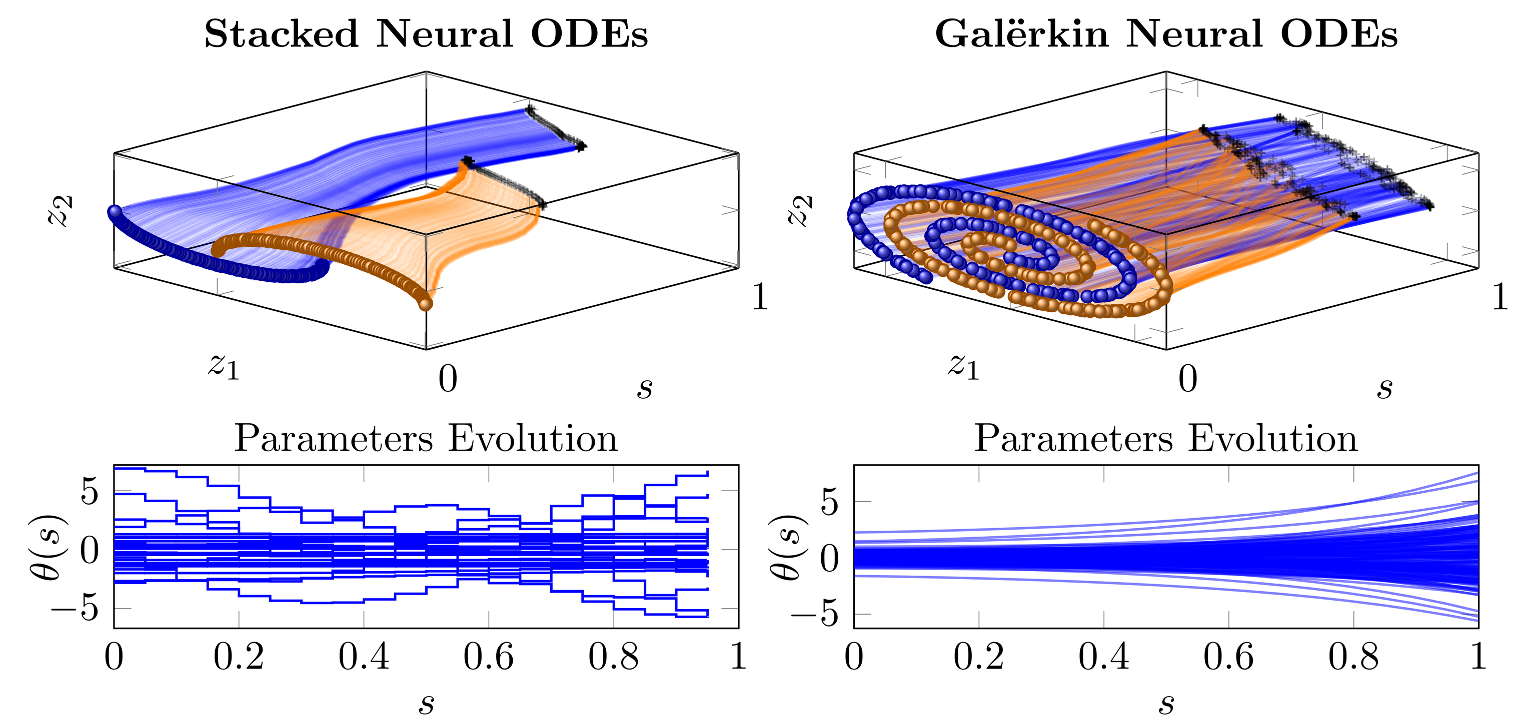
We propose two discretizations:

Spectral (Galërkin)	Depth (stacked)
$\theta(s) = \sum_{j=1}^{m} \alpha_j \odot \psi_j(s)$	$\theta(s) = \theta_i \;\; \forall s \in [s_i, s_{i+1}]$

- **spectral** gradients: $\frac{\mathrm{d}\ell}{\mathrm{d}\alpha} = \int_{\mathcal{S}} \mathbf{a}^\top(\tau)\frac{\partial f_{\theta(s)}}{\partial \theta(s)}\psi(\tau)\mathrm{d}\tau$
- **stacked** inference:

$$\mathbf{z}(S) = h_x(\mathbf{x}) + \sum_{i=0}^{p-1}\int_{s_1}^{s_{i+1}} f_{\theta_i}(\tau, \mathbf{z}(\tau))\mathrm{d}\tau$$

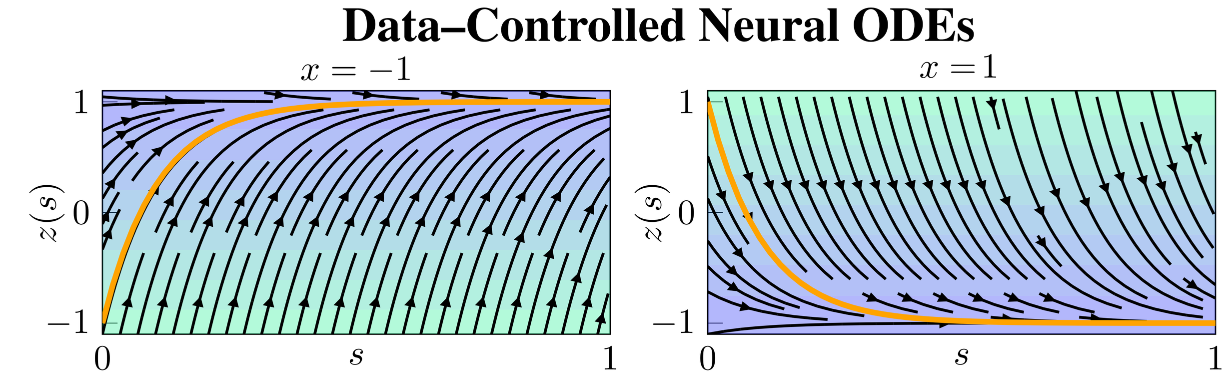
Depth–Variant Neural ODEs



Data–Controlled Neural ODE

Augmentation strategies are not always necessary for Neural ODEs to solve challenging tasks.

- Classic benchmark of reflection $\varphi(x) = -x$
- Neural ODEs can approximate φ without augmentation **if** the also input x is fed to $f_\theta \Rightarrow$ **data–conditioned** vector field

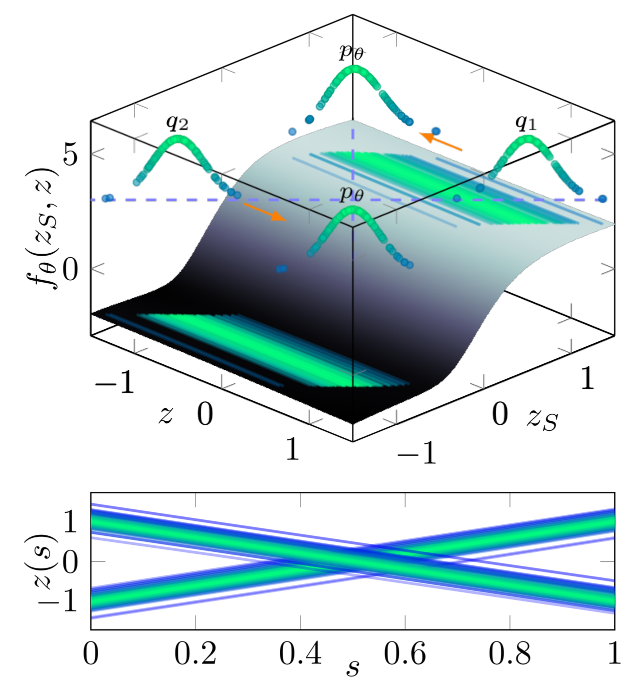


- We can define the general *data–controlled* Neural ODEs:

$$\dot{\mathbf{z}}(s) = f_{\theta(s)}(s, \mathbf{x}, \mathbf{z}(s)), \quad \mathbf{z}(0) = h_x(\mathbf{x}).$$

It learns a **family** of vector fields rather than a single one

Conditional Continuous Normalizing Flows

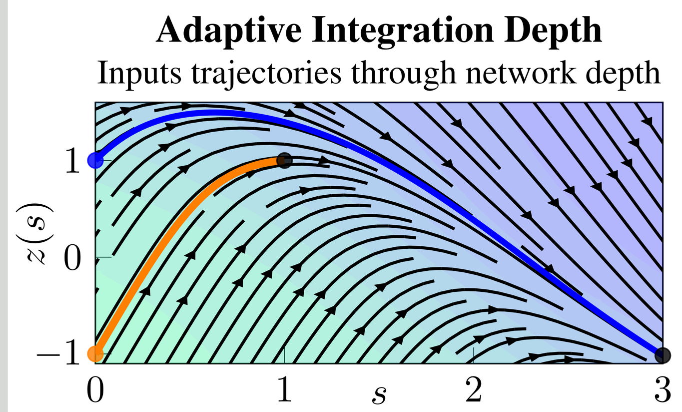


- Data–controlled CNFs can be used in multi–objective generative tasks
- Use a single model to sample from N different distribution p_θ by warping N predetermined known distributions q_i.

e.g. we can learn to conditionally sample from two distributions:

$$\dot{z}(s) = f_\theta(z_S, z(s))$$
$$z(S) = z_S, \quad z_S \sim q_1 \text{ or } z_S \sim q_2$$

Adaptive–Depth Neural ODE



- $\varphi(x)$ can be learned without the need of any *crossing trajectory*.
- If each input is integrated in a different depth domain $\mathcal{S}(x)$, no crossing flows are needed.
- A **hypernetwork** $g_\omega : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\omega} \to \mathbb{R}$ can be trained to learn the integration depth of each sample.

We define the general *adaptive depth* class as Neural ODEs performing the mapping $\mathbf{x} \mapsto \phi_{g_\omega(\mathbf{x})}(\mathbf{x})$, i.e.

$$\hat{\mathbf{y}} = h_y\left(h_x(\mathbf{x}) + \int_0^{g_\omega(\mathbf{x})} f_{\theta(s)}(\tau, \mathbf{x}, \mathbf{z}(\tau))d\tau\right),$$

https://github.com/DiffEqML

massaroli@robot.t.u-tokyo.ac.jp, poli_m@kaist.ac.kr