Hypersolvers: Toward Fast Continuous–Depth Models

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

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

Continuous–Depth Models

We consider the following general Neural ODE formulation:

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

Neural ODEs are Slow

- **Known**: continuous–depth models are too slow for meaningful **large-scale** or **embedded** applications
- **Existing methods:** regularization terms or augmentation can be used to reduce stiffness and improve inference speed. However, in many settings the vector field $f_{\theta(s)}$ **cannot be modified** e.g control applications.
- We propose a new framework to improve Neural ODE speed in both **inference** and **training**.

A New Paradigm for Neural ODE: `Hypersolvers`

- **Novel paradigm** to analyze **model–solver interplay**.
- Orthogonal to existing regularization approaches.
- **Idea:** improve pareto efficiency by approximating local residuals with neural networks

General `Hypersolver` formulation:

$$\begin{aligned} \mathbf{z}_{k+1} &= \mathbf{z}_k + \overbrace{\epsilon\psi(s_k, \mathbf{x}, \mathbf{z}_k)}^{\text{solver step}} + \epsilon^{p+1}\overbrace{g_\omega(\epsilon, s_k, \mathbf{x}, \mathbf{z}_k)}^{\text{hypersolver net}} \\ \mathbf{z}_0 &= h_x(\mathbf{x}) \\ \hat{\mathbf{y}}_k &= h_y(\mathbf{z}_k) \end{aligned} \quad k \in \mathcal{K}$$

where ψ is the update step of a p-th order solver

`Hypersolver` Training Strategies

Assume to have the *exact* solution of the Neural ODE (practically: use adaptive–step solvers with low tolerances). Train g_ω with **local truncation residuals** as supervised labels:

$$\mathcal{R}(s_k, \mathbf{z}(s_k), \mathbf{z}(s_{k+1})) = \frac{1}{\epsilon^{p+1}}\left[\mathbf{z}(s_{k+1}) - \mathbf{z}(s_k) - \epsilon\psi(s_k, \mathbf{x}, \mathbf{z}(s_k))\right]$$

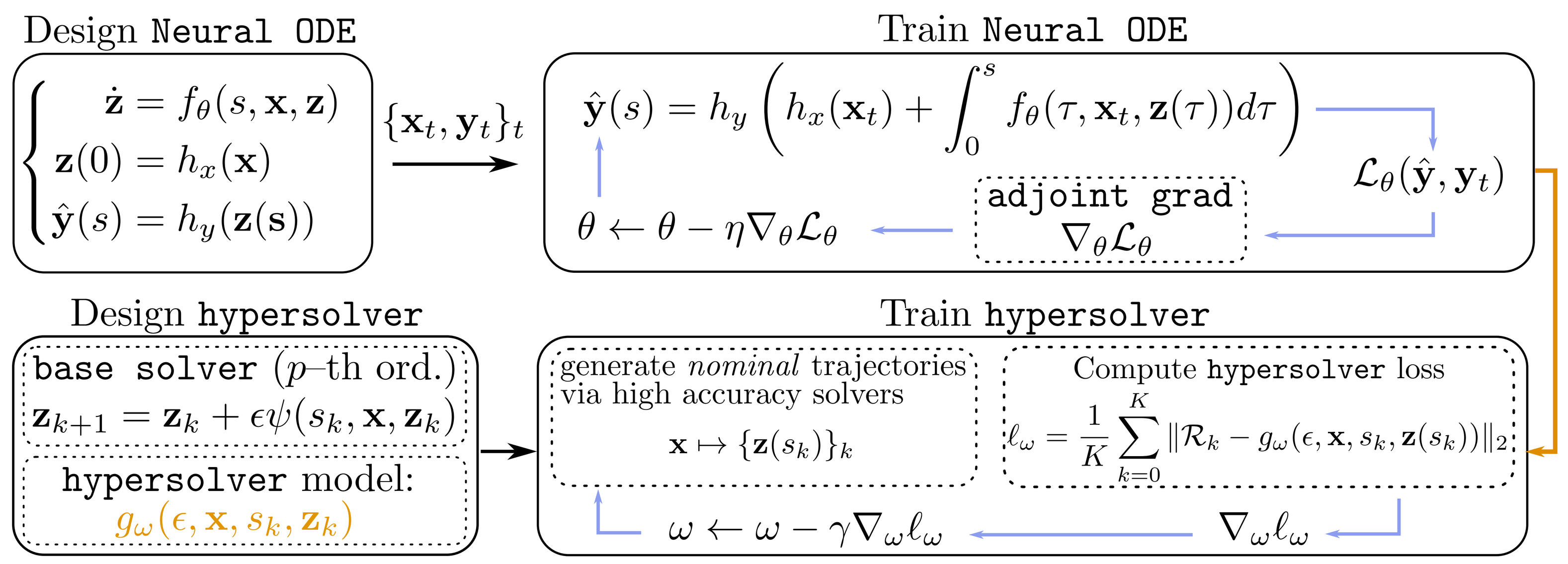
If g_ω is a $\mathcal{O}(\delta)$ approximator of $\mathcal{R}$, i.e.

$$\forall k \in \mathbb{N}_{\leq K} \quad \|\mathcal{R}(s_k, \mathbf{z}(s_k), \mathbf{z}(s_{k+1})) - g_\theta(\epsilon, s_k, \mathbf{x}, \mathbf{z}(s_k))\|_2 \leq \mathcal{O}(\delta),$$

the local truncation error e_k of the hypersolver is $\mathcal{O}(\delta\epsilon^{p+1})$.

Alternative: trajectory fitting and **adversarial training**

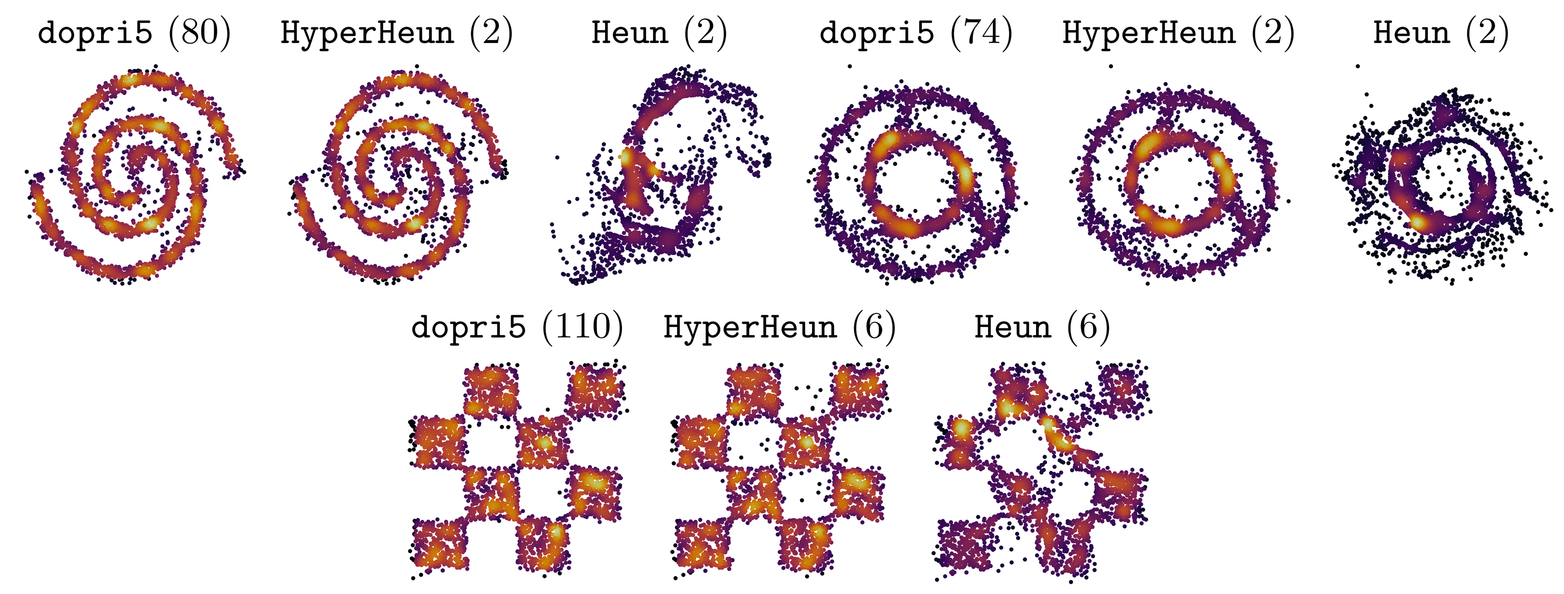
Blueprint for `Hypersolver` Design



A blueprint for hypersolver design

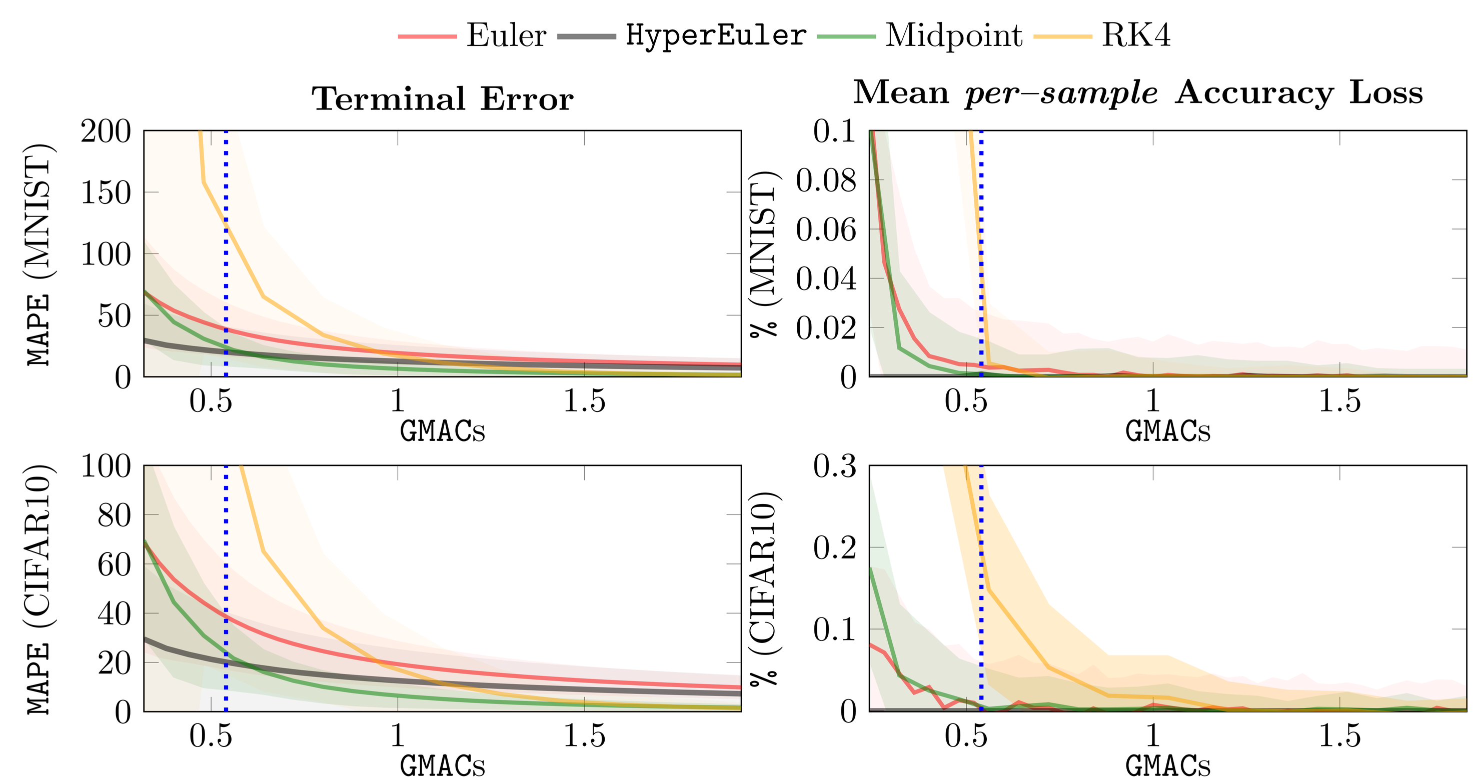
1. design your continuous model & task
2. train your NeuralODE
3. choose a high accuracy solver
4. choose a base solver (ψ)
5. choose a model g_ω for the hypersolver
6. generate nominal trajectories of the Neural ODE via the high accuracy solver
7. train the hypersolver network

Fast Continuous Normalizing Flow Sampling



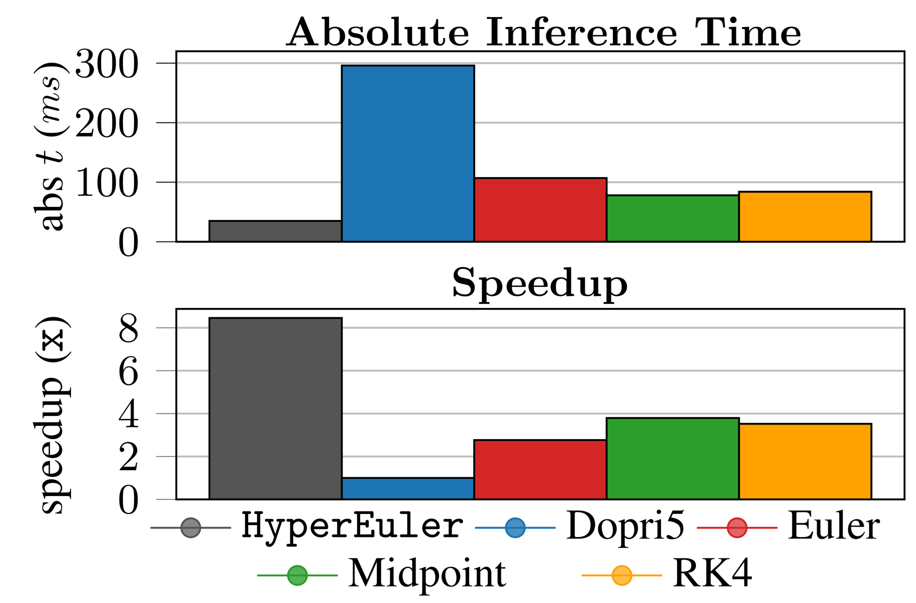
Reconstructed densities (CNFs) solved with various methods. In parenthesis, number of function evaluations. `HyperHeun` preserves sampled quality with low NFEs.

Pareto Efficiency



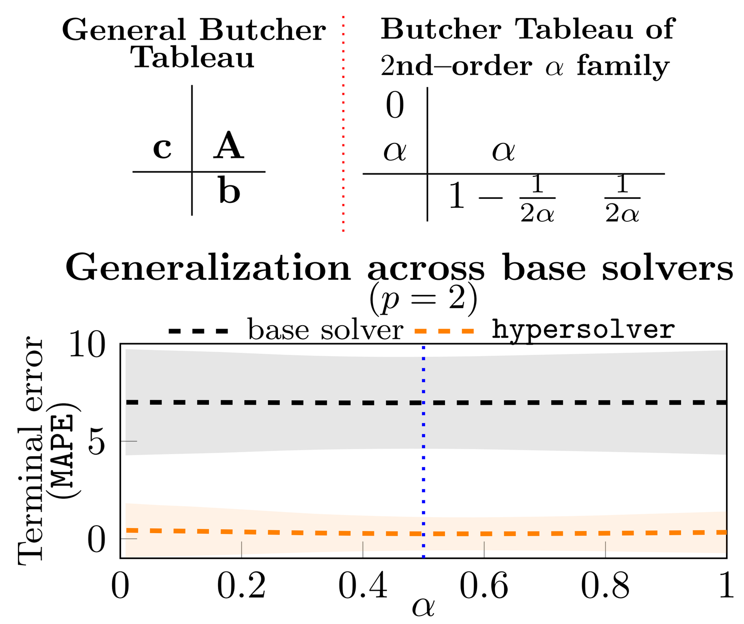
Test accuracy loss %–NFE and MAPE–GMAC Pareto fronts of different ODE solvers on MNIST and CIFAR10 test sets. `HyperEuler` shows higher pareto efficiency for low function evaluations (NFEs) even over higher–order methods.

Wall–Clock Inference Speedups



Absolute time (ms) speedup: fixed–step methods over `dopri5`, MNIST test set. `HyperEuler` solves the Neural ODE 8x **faster** than `dopri5` with the **same accuracy**.

Generalization Across Base Solvers



Hypersolvers **generalize** across **different base solvers of the same order** (second order explicit solver family parametrized by α).

`HyperMidpoint` ($\alpha = 0.5$) generalized without finetuning to other base solvers, preserving its pareto efficiency over the entire α–family.

`Hypersolver` Overhead

- **Overhead of the method**: evaluation of g_ω at each solver step.
- `Hypersolver` computational overheads decreases as the base solver order p increases.

Even in the worst-case scenario (low order), hypersolvers remain pareto efficient over traditional methods

Accelerating Neural ODE Training

- **Challenge:** ensuring that the `hypersolver` network remains a $O(\delta)$ approximator of residuals $\mathcal{R}$ across training iterations.
- Maximizing `hypersolver` reuse across training iterations represents a path toward faster Neural ODEs (even training).

Beyond Fixed–Step `Hypersolvers`

- `Hypersolver` are **not limited** to **fixed–step explicit** solvers.
- Adaptive-step, *predictor–corrector* schemes, different classes of differential equations are also compatible.
- Adversarial training may be used to enhance `hypersolver` resilience to challenging dynamics.

https://github.com/DiffEqML

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