Taylor-Lagrange Neural Ordinary Differential Equations: Towards Fast and Accurate Training of Neural ODEs

Franck Djeumou[1], Cyrus Neary [1], Eric Goubault[2], Sylvie Putot[2], Ufuk Topcu[1]

[1] The University of Texas at Austin, United States

[2] LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, France

autonomous SYSTEMS GROUP

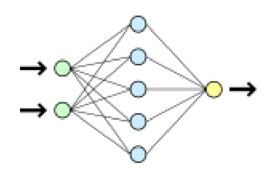
The central question

How can we create a **fast method** to train neural ordinary differential equations (neural ODEs), **without incurring performance losses** in the trained model?

What are neural ODEs?

A class of deep learning models that uses neural networks to parametrize differential equations.

$$\dot{x}(t) = \boxed{f_\theta(x(t))}$$



How do we evaluate neural ODEs?

Numerically solve the differential equation parametrized by the neural network $f_\theta(\cdot)$.

i.e. Given initial state x and prediction time T, solve:

$$NeuralODE_\theta(x, T) = x + \int_0^T f_\theta(x(s))ds$$

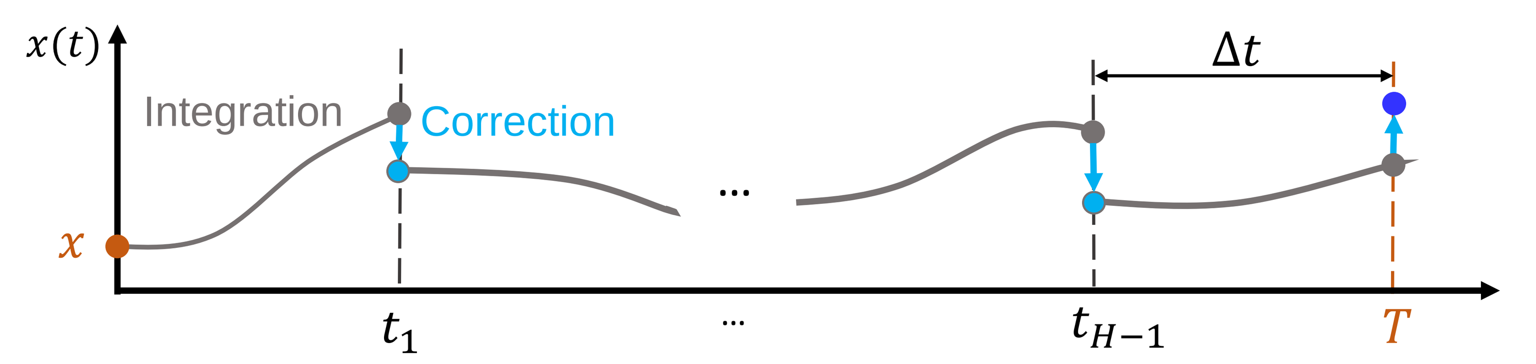
The challenges of training Neural ODEs

Training Neural ODEs is **computationally expensive:**

- Model evaluations requires **numerical integration.**
- Parameter updates require **gradient computations through the ODE solution.**
- Empirically, numerical integration becomes **more challenging as training progresses.**

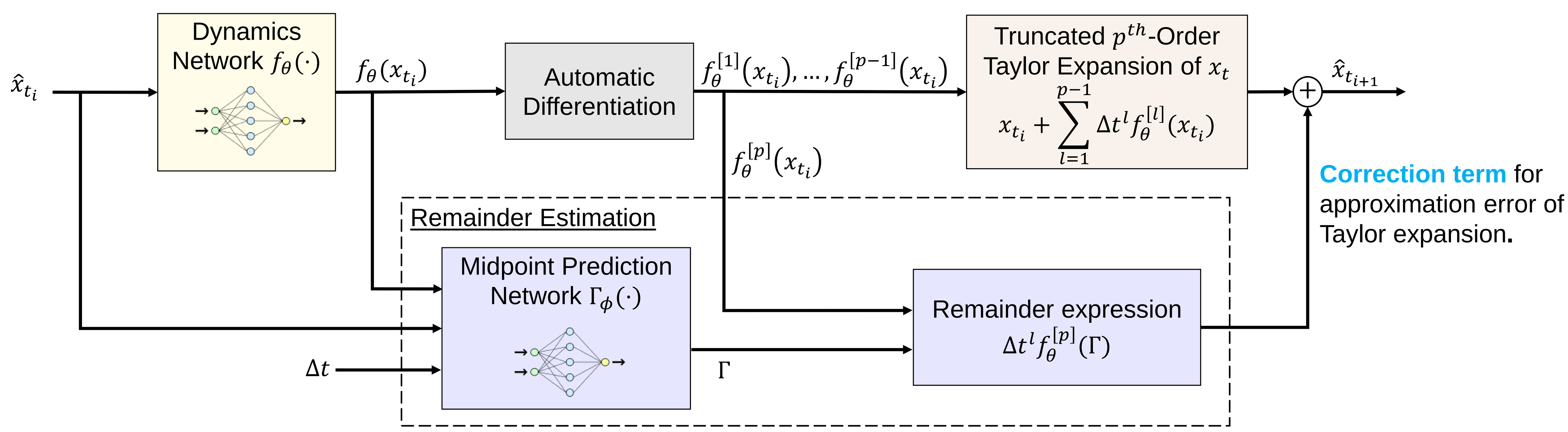
A summary of the approach

1. Use fixed timestep integration methods with a coarse temporal discretization to **quickly obtain approximate evaluations**.
2. Use a learned model to **correct for the introduced integration errors**.



An Illustration of the Approach

$$\hat{x}_{t_{i+1}} = \underbrace{\hat{x}_{t_i} + \Delta t f_\theta(\hat{x}_{t_i}) + \cdots + \Delta t^{p-1} f_\theta^{[p-1]}(\hat{x}_{t_i})}_{\text{Truncated } p^{th}\text{-Order Taylor Expansion of } x_t} + \underbrace{\Delta t^p f_\theta^{[p]}\left(\hat{x}_{t_i} + \Gamma(\hat{x}_{t_i}, \Delta t) f_\theta(\hat{x}_{t_i})\right)}_{\text{Correction term for expansion error}}$$



Correction term for approximation error of Taylor expansion.

Training $f_\theta(\cdot)$ and $\Gamma_\phi(\cdot)$:
Iterate between steps 1 and 2.

1. Fix $\Gamma_\phi \leftarrow \Gamma_{\hat{\phi}}$ and **train $f_\theta(\cdot)$** to fit the available training data. Use the **correction term's magnitude as a regularizer**.

$$\min_\theta \left[Loss(Model(\theta, \hat{\phi}), TrainData) + \lambda \left\| \Delta t^p f_\theta^{[p]}\left(\Gamma_{\hat{\phi}}\right) \right\|^2 \right]$$

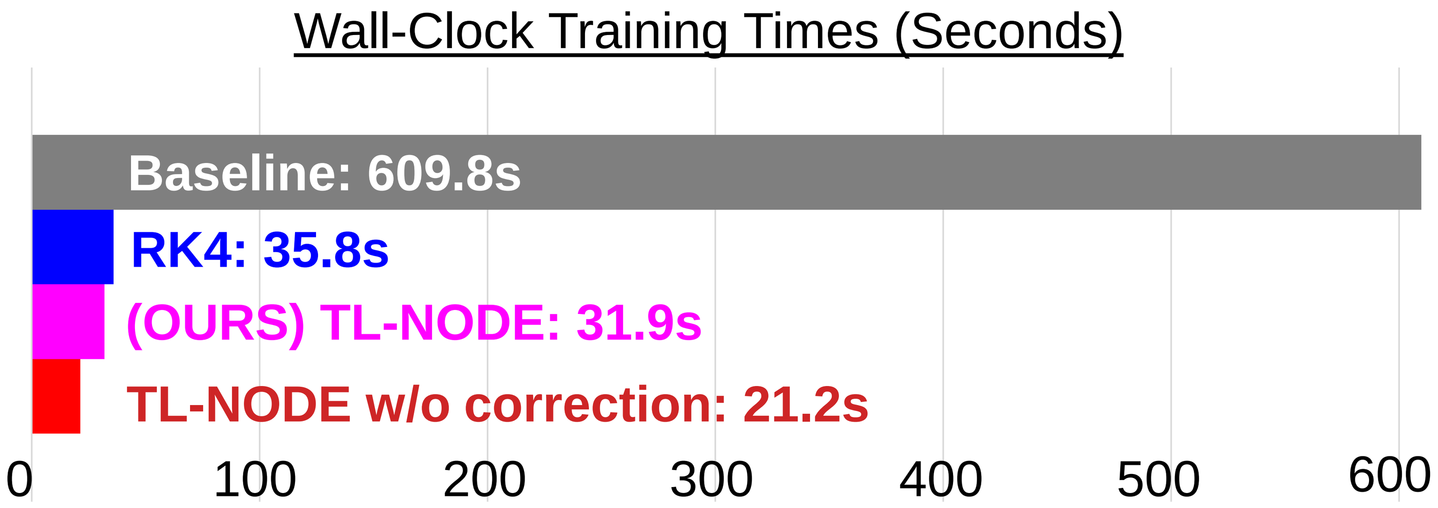
2. Fix $f_\theta \leftarrow f_{\hat{\theta}}$ and **train $\Gamma_\phi(\cdot)$** on data generated using **a high-fidelity ODE solver** to integrate $f_{\hat{\theta}}(\cdot)$.

$$\min_\phi Loss(Model(\hat{\theta}, \phi), HighFidelitySolver(f_{\hat{\theta}}))$$

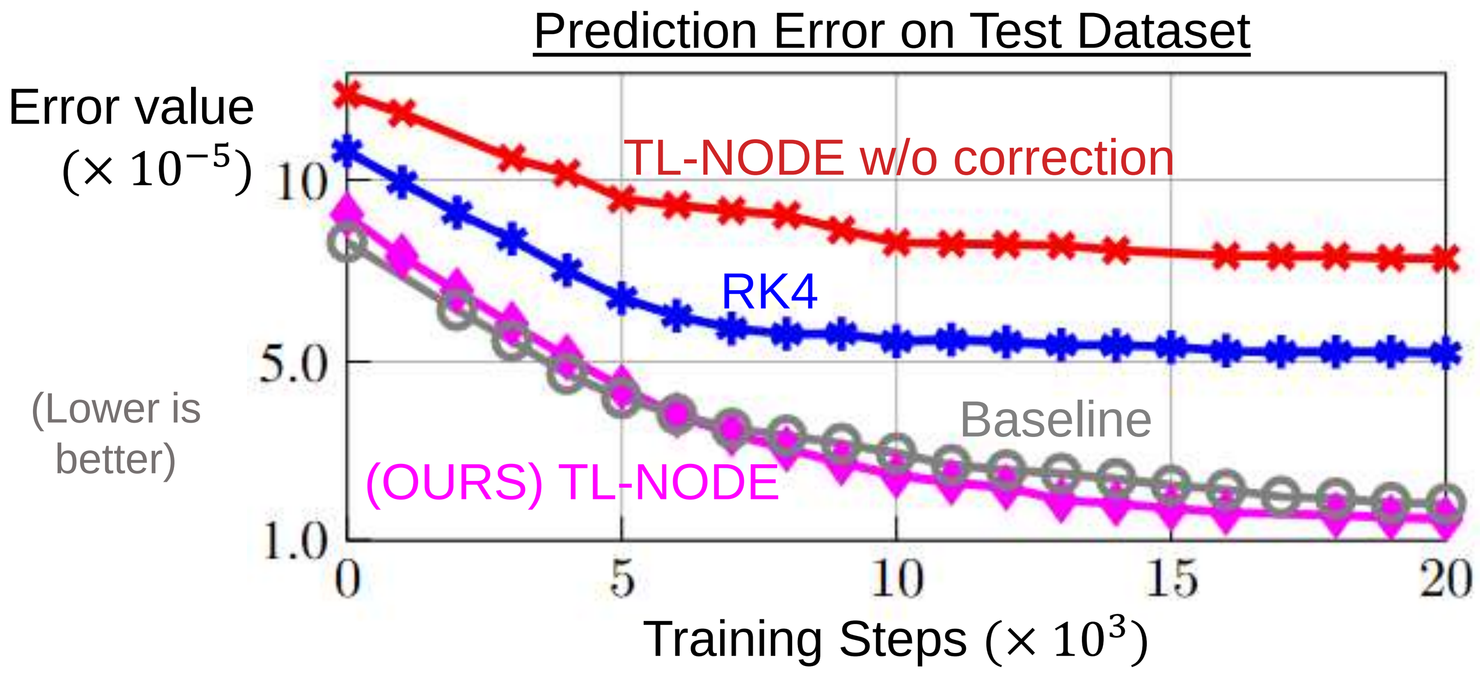
Learning to predict unknown stiff dynamics

TL-NODE is **20x faster** than the baseline, and **5x more accurate** than other fixed-timestep methods.

Wall-Clock Training Times (Seconds)



Prediction Error on Test Dataset

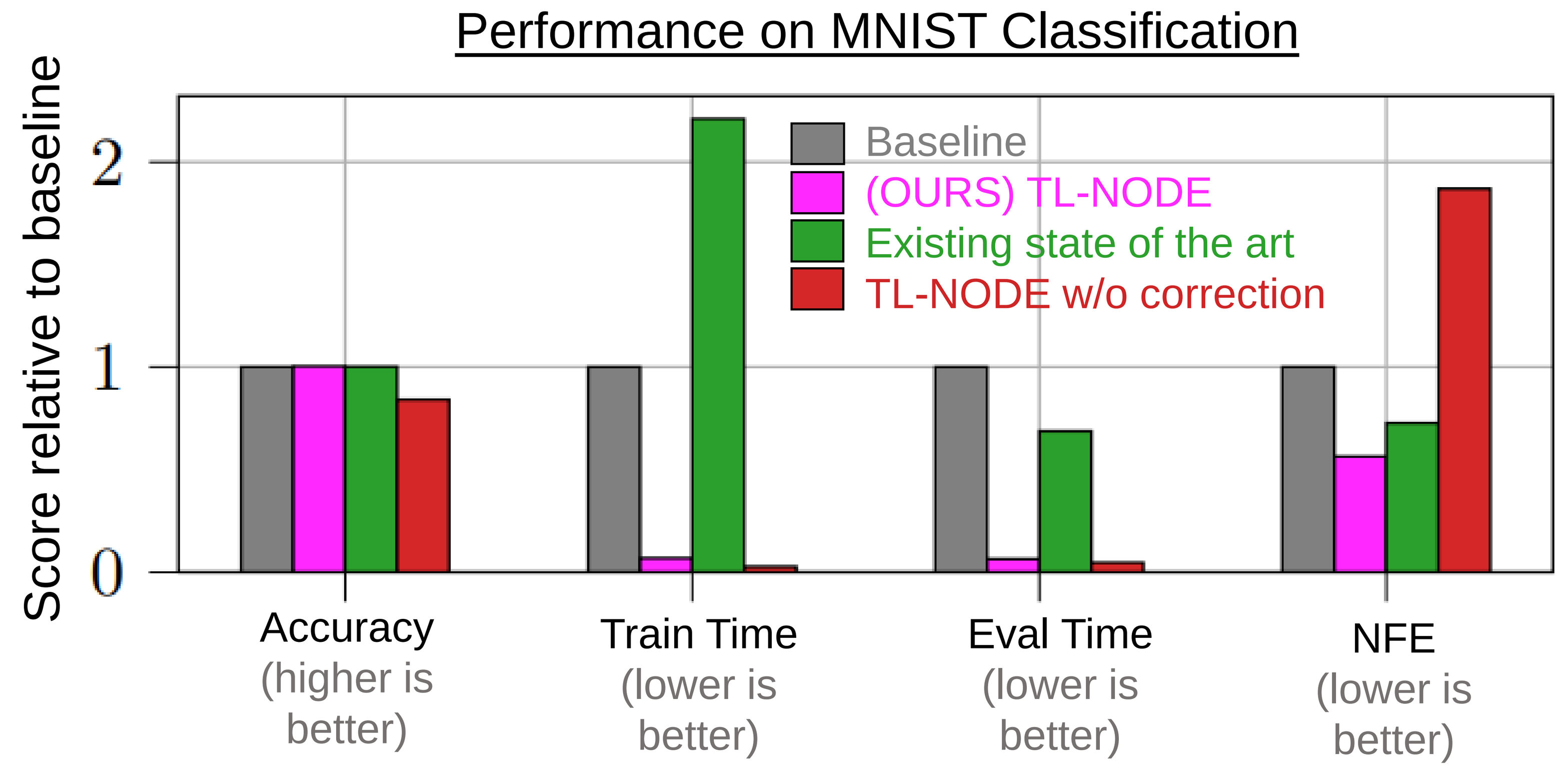


Supervised classification task

Training and evaluating TL-NODEs is **16x faster than the baseline method** while enjoying the **same level of accuracy**.

TL-NODEs requires the **fewest number of evaluations (NFE) of $f_\theta(\cdot)$** per solution of the neural ODE.

Performance on MNIST Classification



Contact: fdjeumou@utexas.edu, cneary@utexas.edu | © C. Neary, 2022