Istituto Dalle Molle di studi sull'intelligenza artificiale
USI/SUPSI
IDSIA

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

Neural Differential Equations for ***Learning to Program Neural Nets Through Continuous Learning Rules***

Kazuki Irie

Francesco Faccio

Jürgen Schmidhuber

Background: Neural ODEs for Sequences

Neural Controlled DEs (NCDEs) Kidger et al. NeurIPS 2020

- \+ Elegant extension of Neural ODEs for Sequence Processing
- \+ Good empirical performance for supervised time series classification
- \- Only one architecture available, akin to the “vanilla RNN”
- \- Scalability limitation: requires a $\mathbb{R}^d \to \mathbb{R}^{d \times d_{\text{in}}}$ mapping NN

$$\boldsymbol{h}(t) = \boldsymbol{h}(t_0) + \int_{s=t_0}^{t} \boldsymbol{F}_\theta(\boldsymbol{h}(s))d\boldsymbol{x}(s) = \boldsymbol{h}(t_0) + \int_{s=t_0}^{t} \boldsymbol{F}_\theta(\boldsymbol{h}(s))\boldsymbol{x}'(s)ds$$

$\boldsymbol{F}_\theta(\boldsymbol{h}(s)) \in \mathbb{R}^{d \times d_{\text{in}}}$ *!!*

Hidden state $\boldsymbol{h}(t) \in \mathbb{R}^d$ *evolves as a function of* $\boldsymbol{x}(t) \in \mathbb{R}^{d_{\text{in}}}$

Background: Fast Weight Programmers (FWPs)

- NNs that learn to **program** other NNs by **generating rapid weight changes**
- General-purpose auto-regressive sequence processor
- **Outer product version: Transformers with linearised self-attention**
- Sequential **dynamics** based on weight update rules/programming instructions/**learning rules** that have a **straightforward ODE counterpart**

At step n ***Input*** $\boldsymbol{x}_n \in \mathbb{R}^{d_{\text{in}}}$ ***Output*** $\boldsymbol{y}_n \in \mathbb{R}^{d_{\text{out}}}$

$$\begin{aligned} \beta_n, \boldsymbol{q}_n, \boldsymbol{k}_n, \boldsymbol{v}_n &= \boldsymbol{W}_{\text{slow}} \boldsymbol{x}_n \\ \boldsymbol{W}_n &= \boldsymbol{W}_{n-1} + \sigma(\beta_n)(\boldsymbol{v}_n - \boldsymbol{W}_{n-1}\phi(\boldsymbol{k}_n)) \otimes \phi(\boldsymbol{k}_n) \\ \boldsymbol{y}_n &= \boldsymbol{W}_n \phi(\boldsymbol{q}_n) \end{aligned}$$

"New = Old + Update"

General Idea

Discrete-Time Weight Update Equation

$$\beta_n, \boldsymbol{q}_n, \boldsymbol{k}_n, \boldsymbol{v}_n \quad = \quad \boldsymbol{W}_{\text{slow}} \boldsymbol{x}_n$$

$$\boldsymbol{W}_n \quad = \quad \boldsymbol{W}_{n-1} + \sigma(\beta_n)(\boldsymbol{v}_n - \boldsymbol{W}_{n-1}\phi(\boldsymbol{k}_n)) \otimes \phi(\boldsymbol{k}_n)$$

Continuous-Time Counterpart

$$\boldsymbol{W}(t) = \boldsymbol{W}(t_0) + \int_{s=t_0}^{t} \boldsymbol{F}_\theta(\boldsymbol{W}(s), \boldsymbol{x}(s))ds$$

This work

We introduce continuous-time counterparts of linear Transformers/FWPs that

- **Can directly replace existing Neural CDEs/ODEs for sequence processing**
- **Conceptually scale better than existing NCDEs**
- **Empirically outperform existing Neural ODE/CDE models**

We propose multiple model variations, depending on

- ***Smoothness of input control signals***
- **and compare different *learning rule parameterisations (Hebb, Oja, Delta)***

Example: Neural CDE based FWPs

Assume **differentiable** control signals $\boldsymbol{x} : t \mapsto \boldsymbol{x}(t) \in \mathbb{R}^{d_{\text{in}}}$

$$\boldsymbol{W}(t) = \boldsymbol{W}(t_0) + \int_{s=t_0}^{t} \mathbf{F}_\theta(\boldsymbol{W}(s), \boldsymbol{x}(s), \boldsymbol{x}'(s))\boldsymbol{x}'(s)ds$$

$$= \sigma(\beta(s)) \begin{cases} \boldsymbol{W}_k\boldsymbol{x}(s) \otimes \boldsymbol{W}_v\boldsymbol{x}'(s) & \text{Hebb} \\ (\boldsymbol{W}_k\boldsymbol{x}(s) - \boldsymbol{W}(s)^\top \boldsymbol{W}_v\boldsymbol{x}'(s)) \otimes \boldsymbol{W}_v\boldsymbol{x}'(s) & \text{Oja} \\ (\boldsymbol{W}_v\boldsymbol{x}(s) - \boldsymbol{W}(s)\boldsymbol{W}_k\boldsymbol{x}'(s)) \otimes \boldsymbol{W}_k\boldsymbol{x}'(s) & \text{Delta} \end{cases}$$

- ***Scalable***: outer product based vector field
- ***Expressive:*** First: sum rank-1 updates over time (above), then: output:

$$\boldsymbol{y}(T) = \begin{cases} \boldsymbol{W}(T)^\top \boldsymbol{W}_q\boldsymbol{x}(T) & \text{Hebb and Oja} \\ \boldsymbol{W}(T)\boldsymbol{W}_q\boldsymbol{x}'(T) & \text{Delta} \end{cases}$$

*(vs. **basic NCDEs** with a rank-1 vector field: **scalable but not expressive**)*

Speech Commands & PhysioNet Sepsis

Type	Model	Speech Commands	PhysioNet Sepsis	
			OI	no-OI
Direct NODE	GRU-ODE [4]*	47.9 (2.9)	85.2 (1.0)	77.1 (2.4)
	Hebb	82.8 (1.1)	**90.4 (0.4)**	82.9 (0.7)
	Oja	**85.4 (0.9)**	88.9 (1.4)	82.9 (0.5)
	Delta	81.5 (3.8)	89.8 (1.0)	**84.5 (2.9)**
CDE	NCDE [4]*	89.8 (2.5)	88.0 (0.6)	77.6 (0.9)
	Hebb	89.5 (0.3)	89.9 (0.6)	**85.7 (0.3)**
	Oja	90.0 (0.7)	**91.2 (0.4)**	85.1 (2.5)
	Delta	**90.2 (0.2)**	90.9 (0.2)	84.5 (0.7)

FWP variants largely outperform the NODE baseline
Also obtain slight improvements over the baseline in the NCDE case

Speech Commands & PhysioNet Sepsis

Type	Model	Speech Commands	PhysioNet Sepsis	
			OI	no-OI
Direct NODE	GRU-ODE [4]*	47.9 (2.9)	85.2 (1.0)	77.1 (2.4)
	Hebb	82.8 (1.1)	**90.4 (0.4)**	82.9 (0.7)
	Oja	**85.4 (0.9)**	88.9 (1.4)	82.9 (0.5)
	Delta	81.5 (3.8)	89.8 (1.0)	**84.5 (2.9)**
CDE	NCDE [4]*	89.8 (2.5)	88.0 (0.6)	77.6 (0.9)
	Hebb	89.5 (0.3)	89.9 (0.6)	**85.7 (0.3)**
	Oja	90.0 (0.7)	**91.2 (0.4)**	85.1 (2.5)
	Delta	**90.2 (0.2)**	90.9 (0.2)	84.5 (0.7)

Particularly large improvements in the challenging no-OI setting
Overall, no clear winner among different learning rules for these tasks

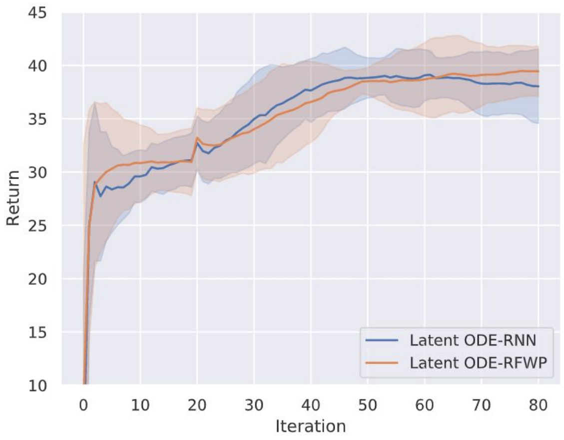
EigenWorms (very long sequences)

Model	Sig-Depth	Step	Test Acc. [%]
NRDE [37]*	2	4	83.8 (3.0)
Hebb	2	4	45.6 (5.9)
Oja			46.7 (7.5)
Delta			**87.7 (1.9)**
NCDE [37]*	1	4	66.7 (11.8)
Hebb	1	4	41.0 (6.5)
Oja			49.7 (9.9)
Delta			**91.8** (3.4)

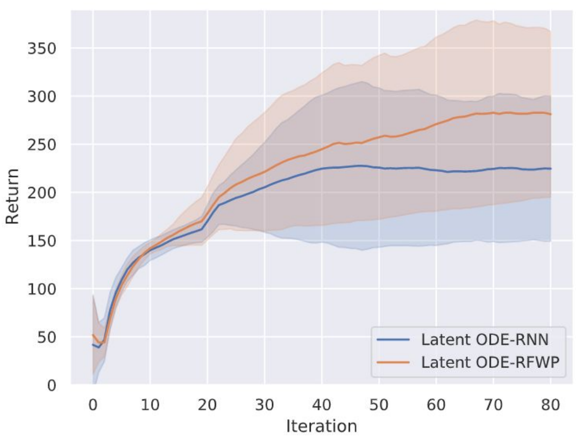
Very large improvements over the best existing ODE based model (NRDE)
Delta rule clearly outperforms others

Model based RL, MuJoCo

We also propose FWP analogs to Latent ODE-RNNs that work as well or better than Latent ODE-RNNs



(a) Swimmer



(b) Hopper

Treat the case where we need to directly work with discrete observations

Experiments in the appendix:

MuJoCo with action repetitions / semi-MDP settings

Thank you for your attention.

https://github.com/IDSIA/neuraldiffeq-fwp

Hope to see you at the poster!