Graph Neural Ordinary Differential Equations

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

[1]Korea Advanced Institute of Technology (KAIST),
[2]The University of Tokyo,
[⋆]Equal contribution authors

The First International Workshop on Deep Learning on Graphs: Methodologies and Applications (DLGMA'20)
2020-02–08

Continuous–depth Learning

Standard DL Settings	**DL on Graphs**
$\mathbf{h}_{s+1} = \mathbf{h}_s + \mathbf{f}(\mathbf{h}_s, \theta_s)$	$\mathbf{H}_{s+1} = \mathbf{H}_s + \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_s\Theta_s)$
residual layer	*graph convolution layer*

Objective: develop the **continuous–depth** paradigm for deep learning.

Neural ODEs	**Graph Neural ODEs**
$\dot{\mathbf{h}}(s) = \mathbf{f}(s, \mathbf{h}(s), \theta)$	$\dot{\mathbf{H}}(s) = \mathbf{F}_{\mathcal{G}}(s, \mathbf{H}(s), \Theta)$
[R. T. Chen et al., 2018]	**proposed approach**

Graph Neural Ordinary **D**ifferential **E**quations (**GDEs**) blend **discrete topological structures** and **differential equations**.

Continuous–depth Learning

Standard DL Settings	**DL on Graphs**
$\mathbf{h}_{s+1} = \mathbf{h}_s + \mathbf{f}(\mathbf{h}_s, \theta_s)$	$\mathbf{H}_{s+1} = \mathbf{H}_s + \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_s\Theta_s)$
residual layer	*graph convolution layer*

Objective: develop the **continuous–depth** paradigm for deep learning.

Neural ODEs	**Graph Neural ODEs**
$\dot{\mathbf{h}}(s) = \mathbf{f}(s, \mathbf{h}(s), \theta)$	$\dot{\mathbf{H}}(s) = \mathbf{F}_{\mathcal{G}}(s, \mathbf{H}(s), \Theta)$
[R. T. Chen et al., 2018]	**proposed approach**

Graph Neural Ordinary **D**ifferential **E**quations (**GDEs**) blend **discrete topological structures** and **differential equations**.

Continuous–depth Learning

Standard DL Settings	**DL on Graphs**
$\mathbf{h}_{s+1} = \mathbf{h}_s + \mathbf{f}(\mathbf{h}_s, \theta_s)$	$\mathbf{H}_{s+1} = \mathbf{H}_s + \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_s\Theta_s)$
residual layer	*graph convolution layer*

Objective: develop the **continuous–depth** paradigm for deep learning.

Neural ODEs	**Graph Neural ODEs**
$\dot{\mathbf{h}}(s) = \mathbf{f}(s, \mathbf{h}(s), \theta)$	$\dot{\mathbf{H}}(s) = \mathbf{F}_{\mathcal{G}}(s, \mathbf{H}(s), \Theta)$
[R. T. Chen et al., 2018]	**proposed approach**

Graph Neural Ordinary **D**ifferential **E**quations (**GDEs**) blend **discrete topological structures** and **differential equations**.

GDEs blend **discrete topological structures** and **differential equations**.

Advantages:

- **Static settings:** computational advantages by incorporation of **numerical methods** in the forward pass.
- **Dynamic settings:** exploitation of the **geometry of the underlying dynamics** and flexibility with respect to **irregular observations**.

Notation:

set of nodes	set of edges	graph	features
$\mathcal{V}$ $(\lvert\mathcal{V}\rvert = n)$	$\mathcal{E} \subset \{(u,v)\}_{u,v\in\mathcal{V}}$	$\mathcal{G} := (\mathcal{V}, \mathcal{E})$	$\mathbf{X} \in \mathbb{R}^{n\times d}$

Graph Neural ODEs (GDEs)

Graph Neural Networks

$$\begin{cases} \mathbf{H}_{s+1} = \mathbf{H}_s + \mathbf{F}_{\mathcal{G}}\left(s, \mathbf{H}_s, \mathbf{\Theta}_s\right) \\ \mathbf{H}_0 = \mathbf{X}_e \end{cases}, \quad s \in \mathbb{N}$$

where $\mathbf{F}$ is a matrix–valued nonlinear function conditioned on graph $\mathcal{G}$ and $\mathbf{\Theta}_s$ is the tensor of **trainable parameters** of the s-th layer.

Graph Neural ODEs (GDEs) [Proposed]

$$\begin{cases} \dot{\mathbf{H}}_s = \mathbf{F}_{\mathcal{G}}\left(s, \mathbf{H}_s, \mathbf{\Theta}\right) \\ \mathbf{H}_0 = \mathbf{X}_e \end{cases}, \quad s \in \mathcal{S} \subset \mathbb{R} \tag{1}$$

where $\mathbf{F} : \mathcal{S} \times \mathbb{R}^{n \times d} \times \mathbb{R}^{p} \to \mathbb{R}^{n \times d}$ is a **depth–varying vector field** defined on $\mathcal{G}$.

What do GDEs learn?

They learn a **graph–conditioned vector field $\mathbf{F}_{\mathcal{G}}$** (parametrized by a GNN) such that:

$$\mathbf{Y} := \mathbf{H}(s) = \underbrace{\mathbf{X}_e}_{\text{input embedding}} + \int_{\mathcal{S}} \mathbf{F}_{\mathcal{G}}(\tau, \mathbf{H}(\tau), \mathbf{\Theta}) d\tau$$

with

$$\mathbf{X}_e := \mathbf{X}\mathbf{W}, \quad \mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} \mathcal{L}$$

Remark:

The *depth variable* s assuming real values brings, in the limit, the map

$$\mathbf{X} \mapsto \mathbf{H}(s) = \mathbf{Y}$$

to resemble a network with infinitely dense layers
$\Rightarrow$ i.e., GDEs are the **deep limit** of GNNs.

Static Models

Graph Convolution Neural Differential Equations (**proposed**)

By choosing $\mathbf{F}_{\mathcal{G}}$ as a simple GCN, we obtain the **GCDE**:

$$\boxed{\frac{d}{ds}\mathbf{H}(s) = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}(s)\boldsymbol{\Theta})}$$

Alternative convolution filters can be employed instead.

Note:

- With different priors on $\mathbf{F}_{\mathcal{G}}$ we can build other types of GDEs: GAT, diffusive, message passing, etc.
- Moreover, we can include **additional biases** in the model structure, e.g.

second–order:	$\ddot{\mathbf{H}}(s) = \mathbf{F}_{\mathcal{G}}(s, \mathbf{H}(s), \boldsymbol{\Theta})$
stochastic:	$d\mathbf{H}(s) = \mathbf{F}_{\mathcal{G}}(s, \mathbf{H}(s), \boldsymbol{\Theta}_f)ds + \mathbf{G}_{\mathcal{G}}(s, \mathbf{H}(s), \boldsymbol{\Theta}_g)dW$

Semi–supervised Node Classification

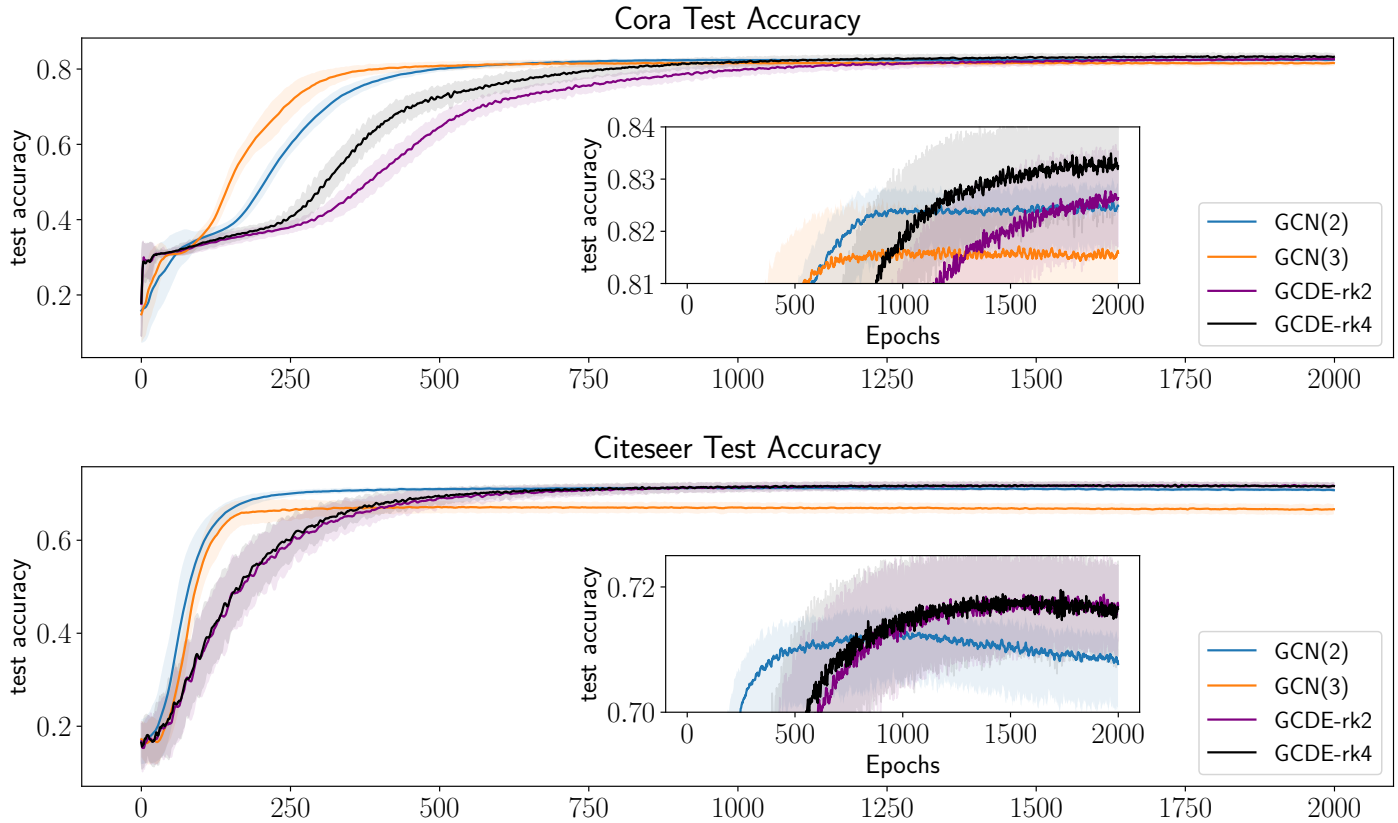
Evaluation on **Cora**, **Citeseer**, **Pubmed**

Ablation study: same **architecture,** different **numerical solvers**

Model (NFE)	Cora	Citeseer	Pubmed
GCN	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$79.0 \pm 0.3\%$
GCN*	$82.8 \pm 0.3\%$	$71.2 \pm 0.4\%$	$79.5 \pm 0.4\%$
GCDE–rk2 (2)	$83.0 \pm 0.6\%$	$72.3 \pm 0.5\%$	$\mathbf{79.9} \pm 0.3\%$
GCDE–rk4 (4)	$\mathbf{83.8} \pm 0.5\%$	$\mathbf{72.5} \pm 0.5\%$	$79.5 \pm 0.4\%$
GCDE–dpr5 **(158)**	$81.8 \pm 1.2\%$	$68.3 \pm 1.2\%$	$78.5 \pm 0.7\%$

Results across 100 runs.

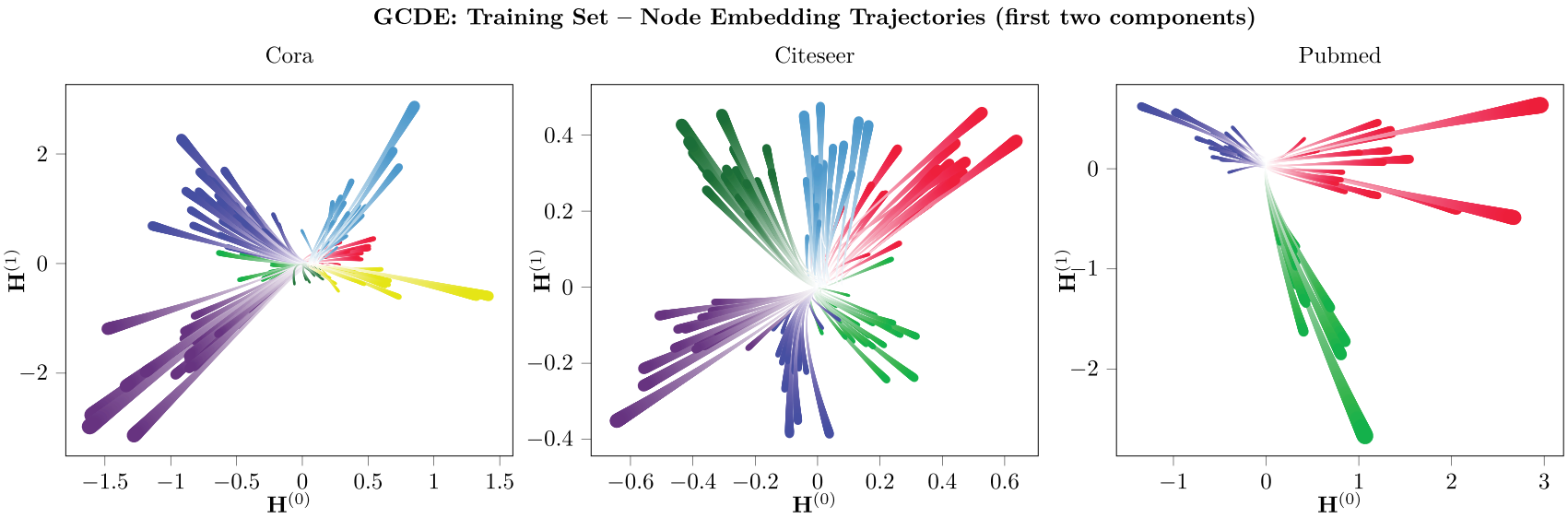
Semi–supervised Node Classification



Higher order solvers: generally more performant, provided graph is dense enough to benefit from additional computation.

No direct advantage of solving the ODE accurately with **adaptive solvers**.

Visualizing Node Feature Trajectories



Trajectories defined by a **forward pass** of GCDE on Cora, Citeseer and Pubmed. Color differentiates between node classes.

The trajectories are divergent, suggesting a **non–decreasing** classification **performance** for GCDE models trained with **longer integration intervals**.

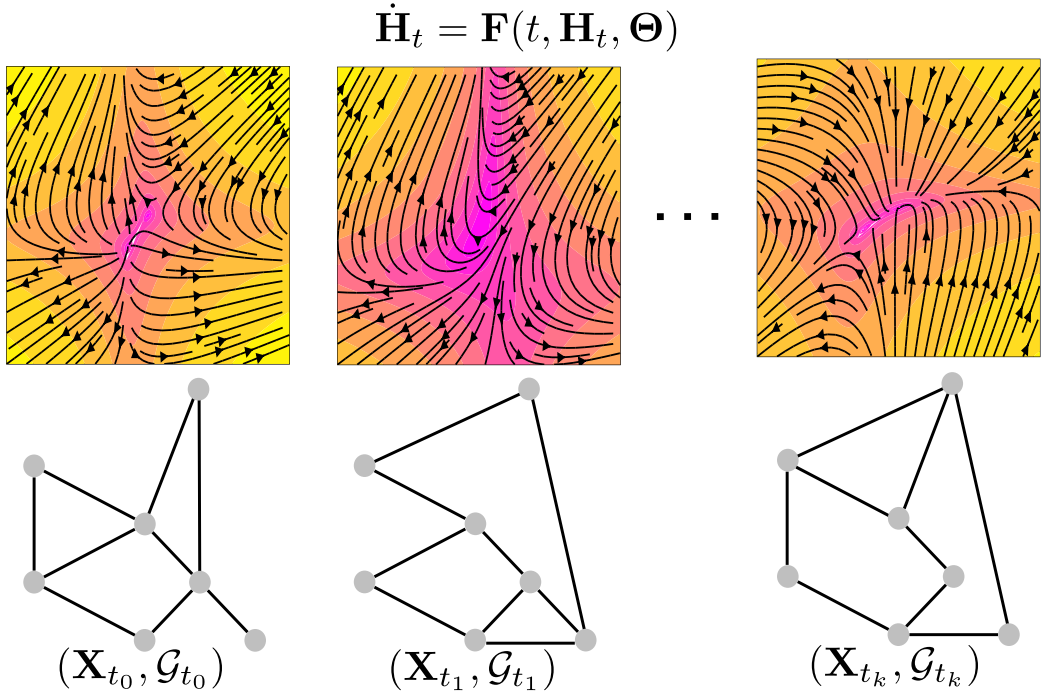
Spatio–Temporal GDEs as Hybrid Systems

In dynamic settings, i.e. direct modeling of dynamical systems, the **depth** variable s assumes the meaning of **time**: $s := t$ and can be modified depending on the requirements.
For example, given a time window Δt, the **prediction** performed by a **GDE** assumes the form:

$$\mathbf{H}(t + \Delta t) = \mathbf{H}(t) + \int_{t}^{t+\Delta t} \mathbf{F}\left(\tau, \mathbf{H}(\tau), \mathbf{\Theta}\right) d\tau,$$

We can extend the GDE framework to settings with sequences of graphs: $\{\mathcal{G}\}$ by leveraging **hybrid dynamical system** machinery.

Spatio–Temporal GDEs as Hybrid Systems



- $\mathcal{T} := \{t_k\}_{k \in \mathcal{K}}$, $\mathcal{K} \subset \mathbb{N} \setminus \{0\}$: set of time instants
- $\{(\mathbf{X}_t, \mathcal{G}_t)\}_{t \in \mathcal{T}}$: *state-graph data stream*:

GDEs models vector fields defined on graphs. Autoregressive GDEs can handle **dynamic topologies** (jumps).

General Autoregressive GDE

The solution of a general autoregressive GDE model (one timestamp):

$$\begin{cases} \dot{\mathbf{H}}(s) & = \mathbf{F}_{\mathcal{G}_{t_k}}(\mathbf{H}(s), \boldsymbol{\Theta}) & s \in [t_{k-1}, t_k] \\ \mathbf{H}^+(s) & = \mathbf{G}_{\mathcal{G}_{t_k}}(\mathbf{H}(s), \mathbf{X}_{t_k}) & s = t_k \qquad k \in \mathcal{K}, \\ \mathbf{Y} & = \mathbf{K}(\mathbf{H}(s)) & s = t_k \end{cases} \tag{2}$$

- $\mathbf{F}, \mathbf{G}, \mathbf{K}$ are GNN–like operators or general neural network layers
- $\mathbf{H}^+$ represent the value of $\mathbf{H}$ after the discrete transition.

Idea

GDEs **smoothly steer** latent node features between two time instants and then apply some discrete operator, resulting in a "**jump**" of $\mathbf{H}$ which is then processed by an output layer.

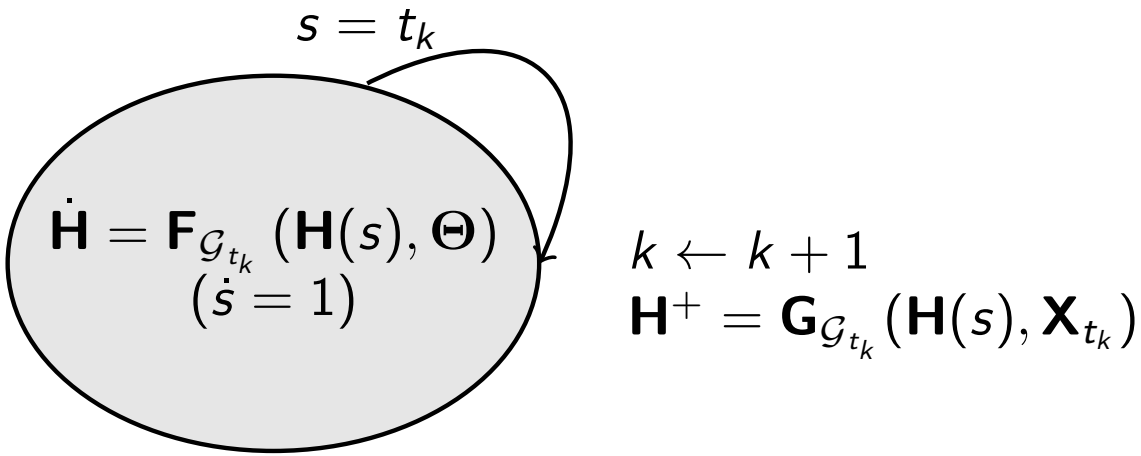


Figure: Schematic of autoregressive GDEs as hybrid automata.

Continuous–depth version of GCGRUs, **GCDE–GRU**:

$$\begin{cases} \dot{\mathbf{H}}(s) & = \mathbf{F}_{\texttt{GCN}}(\mathbf{H}(s), \Theta) & s \in [t_{k-1}, t_k] \\ \mathbf{H}^+(s) & = \texttt{GCGRU}(\mathbf{H}(s), \mathbf{X}_{t_k}) & s = t_k \\ \mathbf{Y} & = \mathbf{K}(\mathbf{H}(s)) & s = t_k \end{cases} \quad k \in \mathcal{K},$$

GCDE–RNNs or GCDE–LSTMs can be obtained in a similar fashion.

Traffic Forecasting

Evaluation on an **undersampled** PeMS(M) traffic dataset: 228 sensor stations

To measure robustness to unevenly sampled datasets we turn **regular** observations (5 minute intervals) into **irregular**:

- 70% probability of removal per point
- offline undersampling of training and test data

Model (depth)	MAPE	NRMSE
GRU	27.52 ± 0.00	1.47 ± 0.00
GCGRU	24.80 ± 0.12	1.44 ± 0.00
GCDE–GRU	23.08 ± 0.11	1.40 ± 0.01

Table: Forecasting test results across 5 runs (mean and standard deviation).

Future extensions

Unknown topology:

- Compatibility with GDEs due to the **algebraic nature** of the relation between the **attention** operator and the **node features**
- If an optimal adaptive graph representation $\mathbf{S}(s, \mathbf{H})$ is obtained via some attentive mechanism:

$$\dot{\mathbf{H}} = \sigma\left(\mathbf{S}\mathbf{H}\boldsymbol{\Theta}\right).$$

Control terms:

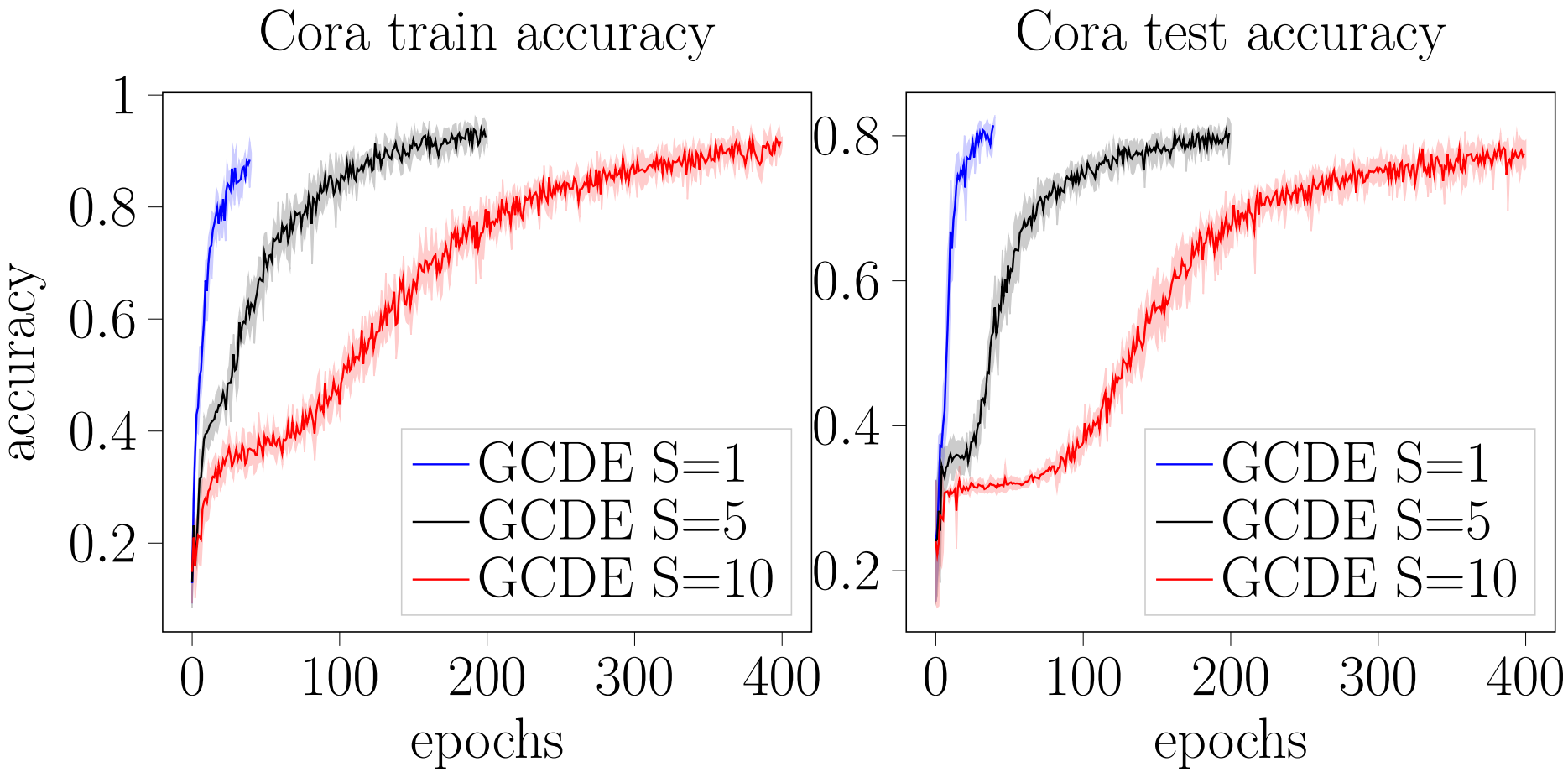
$$\begin{cases} \dot{\mathbf{H}}(s) = \mathbf{F}_{\mathcal{G}}\left(s, \mathbf{H}(s), \boldsymbol{\Theta}\right) + \mathbf{U}\left(s\right) \\ \mathbf{H}(0) = \mathbf{X}_e \end{cases}, \quad s \in \mathcal{S}\,.$$

This approach encompasses a variety of previously proposed approaches, e.g. **special residual connections**. In particular, a choice is $\mathbf{U}\left(s\right) := \mathbf{U}_{\mathcal{G}}\left(s, \mathbf{X}\right)$.

And naturally, other classes of **differential equations**.

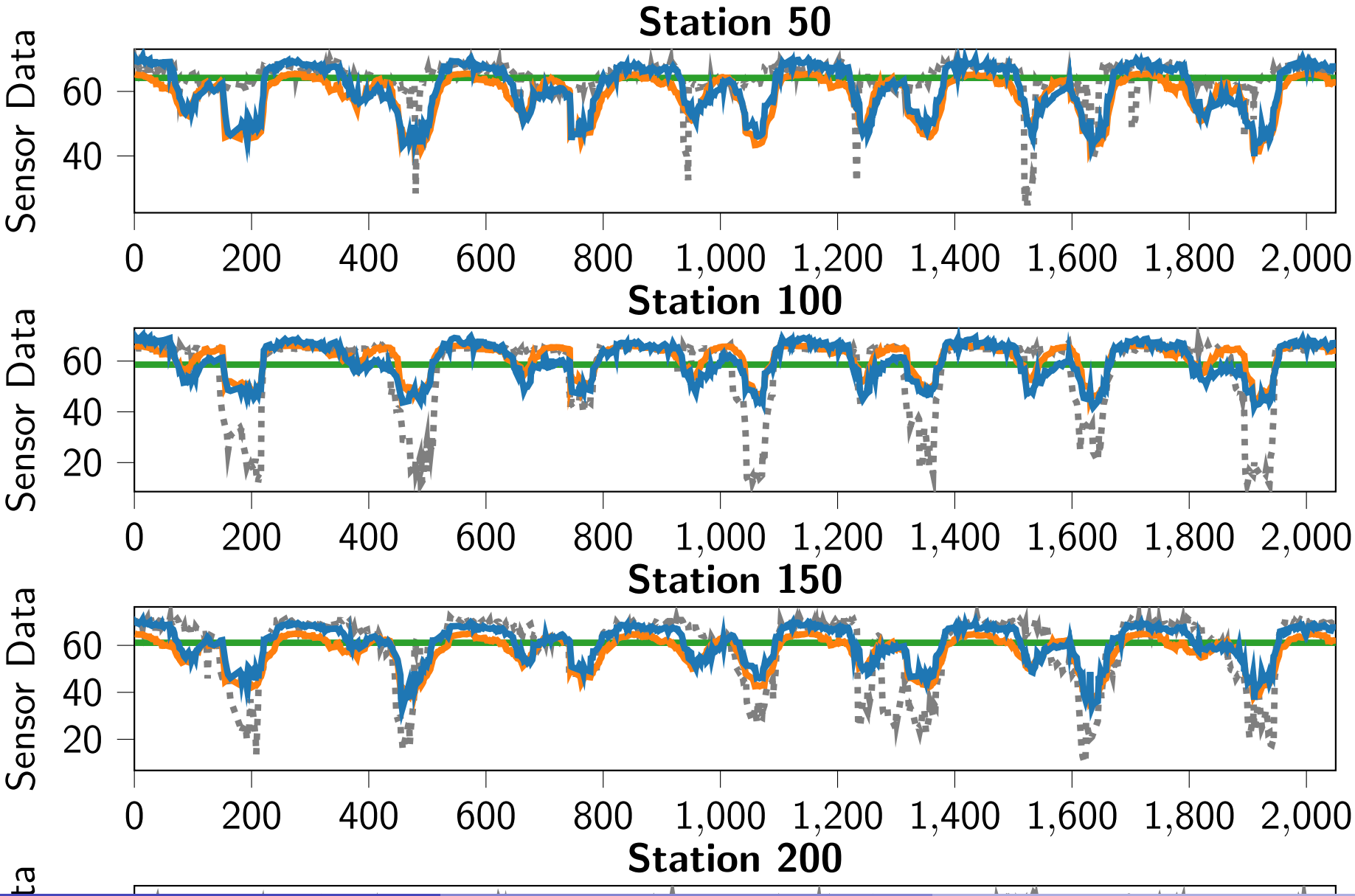
Thank you
Q & A

Appendix



Cora accuracy of GCDE models with **different integration times** s. Higher values of S do not affect performance negatively but require a higher number of epochs.

Appendix


Station 50
Sensor Data
60
40
0
200
400
600
800
1,000
1,200
1,400
1,600
1,800
2,000
Station 100
Sensor Data
60
40
20
0
200
400
600
800
1,000
1,200
1,400
1,600
1,800
2,000
Station 150
Sensor Data
60
40
20
0
200
400
600
800
1,000
1,200
1,400
1,600
1,800
2,000
Station 200