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**
\[ h_{s+1} = h_s + f(h_s, \theta_s) \]
residual layer

**DL on Graphs**
\[ H_{s+1} = H_s + \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H_s \Theta_s) \]
graph convolution layer

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

Neural ODEs
\[ \dot{h}(s) = f(s, h(s), \theta) \]
[R. T. Chen et al., 2018]

Graph Neural ODEs
\[ \dot{H}(s) = F_G(s, H(s), \Theta) \]
proposed approach

**Graph Neural Ordinary Differential Equations (GDEs)** blend **discrete topological structures** and **differential equations**.
**Continuous–depth Learning**

<table>
<thead>
<tr>
<th>Standard DL Settings</th>
<th>DL on Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( h_{s+1} = h_s + f(h_s, \theta_s) )</td>
<td>( H_{s+1} = H_s + \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_s\Theta_s) )</td>
</tr>
<tr>
<td>residual layer</td>
<td>graph convolution layer</td>
</tr>
</tbody>
</table>

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

**Neural ODEs**
\[
\dot{h}(s) = f(s, h(s), \theta)
\]
[R. T. Chen et al., 2018]

**Graph Neural ODEs**
\[
\dot{H}(s) = F_G(s, H(s), \Theta)
\]
proposed approach

**Graph Neural Ordinary Differential Equations (GDEs)** blend **discrete topological structures** and **differential equations**.
Continuous–depth Learning

**Standard DL Settings**

\[ h_{s+1} = h_s + f(h_s, \theta_s) \]

*residual layer*

**DL on Graphs**

\[ H_{s+1} = H_s + \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H_s \Theta_s) \]

*graph convolution layer*

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

**Neural ODEs**

\[ \dot{h}(s) = f(s, h(s), \theta) \]

[R. T. Chen et al., 2018]

**Graph Neural ODEs**

\[ \dot{H}(s) = F_G(s, H(s), \Theta) \]

*proposed approach*

**Graph Neural Ordinary Differential Equations (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:

<table>
<thead>
<tr>
<th>set of nodes</th>
<th>set of edges</th>
<th>graph</th>
<th>features</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{V} (</td>
<td>\mathcal{V} = n</td>
<td>)$</td>
<td>$\mathcal{E} \subset {(u, v)}_{u,v \in \mathcal{V}}$</td>
</tr>
</tbody>
</table>
Graph Neural ODEs (GDEs)

Graph Neural Networks

\[
\begin{cases}
H_{s+1} = H_s + F_G (s, H_s, \Theta_s), \\
H_0 = X_e,
\end{cases}
\quad s \in \mathbb{N}
\]

where \( F \) is a matrix-valued nonlinear function conditioned on graph \( G \) and \( \Theta_s \) is the tensor of \textit{trainable parameters} of the \( s \)-th layer.

\[
\dot{H}_s = F_G (s, H_s, \Theta), \quad s \in S \subset \mathbb{R}
\]

where \( F : S \times \mathbb{R}^{n \times d} \times \mathbb{R}^p \rightarrow \mathbb{R}^{n \times d} \) is a \textit{depth-varying vector field} defined on \( G \).
What do GDEs learn?

They learn a **graph-conditioned vector field** $F_G$ (parametrized by a GNN) such that:

$$
Y := H(s) = \underbrace{X_e}_{\text{input embedding}} + \int_S F_G(\tau, H(\tau), \Theta) d\tau
$$

with

$$
X_e := XW, \quad \Theta^* = \arg\min_\Theta \mathcal{L}
$$

Remark:

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

$$
X \mapsto H(s) = Y
$$

to resemble a network with infinitely dense layers

$\Rightarrow$ i.e., GDEs are the **deep limit** of GNNs.
Graph Convolution Neural Differential Equations (proposed)

By choosing $F_G$ as a simple GCN, we obtain the GCDE:

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

Alternative convolution filters can be employed instead.

Note:

- With different priors on $F_G$ 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: $\ddot{H}(s) = F_G(s, H(s), \Theta)$
  - stochastic: $dH(s) = F_G(s, H(s), \Theta_f) ds + G_G(s, H(s), \Theta_g) dW$
## Semi-supervised Node Classification

### Evaluation on Cora, Citeseer, Pubmed

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

<table>
<thead>
<tr>
<th>Model (NFE)</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>81.4 ± 0.5%</td>
<td>70.9 ± 0.5%</td>
<td>79.0 ± 0.3%</td>
</tr>
<tr>
<td>GCN*</td>
<td>82.8 ± 0.3%</td>
<td>71.2 ± 0.4%</td>
<td>79.5 ± 0.4%</td>
</tr>
<tr>
<td>GCDE–rk2 (2)</td>
<td>83.0 ± 0.6%</td>
<td>72.3 ± 0.5%</td>
<td>79.9 ± 0.3%</td>
</tr>
<tr>
<td>GCDE–rk4 (4)</td>
<td><strong>83.8 ± 0.5%</strong></td>
<td><strong>72.5 ± 0.5%</strong></td>
<td>79.5 ± 0.4%</td>
</tr>
<tr>
<td>GCDE–dpr5 (158)</td>
<td>81.8 ± 1.2%</td>
<td>68.3 ± 1.2%</td>
<td>78.5 ± 0.7%</td>
</tr>
</tbody>
</table>

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 $\Delta t$, the prediction performed by a GDE assumes the form:

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

We can extend the GDE framework to settings with sequences of graphs: $\{G\}$ by leveraging hybrid dynamical system machinery.
Spatio–Temporal GDEs as Hybrid Systems

\[ T := \{ k \in \mathcal{K}, \mathcal{K} \subset \mathbb{N} \setminus \{0\} \} \]: set of time instants

\[ \{(X_t, G_t)\}_{t \in T} \]: 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):

\[
\begin{align*}
\dot{H}(s) &= F_{G_{t_k}}(H(s), \Theta) \quad s \in [t_{k-1}, t_k] \\
H^+(s) &= G_{G_{t_k}}(H(s), X_{t_k}) \quad s = t_k \\
Y &= K(H(s)) \quad s = t_k
\end{align*}
\]  

(2)

- \( F, G, K \) are GNN–like operators or general neural network layers
- \( H^+ \) represent the value of \( H \) 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 \( H \) which is then processed by an output layer.
\[ \dot{H} = F_{g_t} (H(s), \Theta) \quad (s = 1) \]

\[ k \leftarrow k + 1 \]

\[ H^+ = G_{g_t} (H(s), X_{t_k}) \]

**Figure:** Schematic of autoregressive GDEs as hybrid automata.

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

\[
\begin{cases}
\dot{H}(s) &= F_{GCN}(H(s), \Theta) \quad s \in [t_{k-1}, t_k] \\
H^+(s) &= GCGRU(H(s), X_{t_k}) \quad s = t_k \quad k \in K, \\
Y &= K(H(s)) \quad s = t_k
\end{cases}
\]

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

<table>
<thead>
<tr>
<th>Model (depth)</th>
<th>MAPE</th>
<th>NRMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRU</td>
<td>27.52 ± 0.00</td>
<td>1.47 ± 0.00</td>
</tr>
<tr>
<td>GCGRU</td>
<td>24.80 ± 0.12</td>
<td>1.44 ± 0.00</td>
</tr>
<tr>
<td>GCDE–GRU</td>
<td>23.08 ± 0.11</td>
<td>1.40 ± 0.01</td>
</tr>
</tbody>
</table>

**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 $S(s, H)$ is obtained via some attentive mechanism:
  \[
  \dot{H} = \sigma (SH\Theta).
  \]

Control terms:

\[
\begin{aligned}
\dot{H}(s) &= F_G(s, H(s), \Theta) + U(s), \\
H(0) &= X_e, \\
\end{aligned}
\quad s \in S.
\]

This approach encompasses a variety of previously proposed approaches, e.g. special residual connections. In particular, a choice is $U(s) := U_G(s, X)$.

And naturally, other classes of differential equations.
Thank you
Q & A
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.
Traffic data prediction results of 50% undersampling. GCDE–GRUs are able to evolve the latents between timestamps and provide a more accurate fit.