Graph Condensation for Continual Graph Learning

Ruihong Qiu
The University of Queensland
r.qiu@uq.edu.au
Contents

- Continual Graph Learning
- Related Work
- CaT
- PUMA
Contents

- Continual Graph Learning
- Related Work
- CaT
- PUMA
Continual Graph Learning (CGL)

• Traditional Graph Neural Networks (GNNs) are not good at **streaming** inputs.

• **New** nodes can appear **dynamically**
Class/Task Incremental Learning (IL)

- **Class-IL** is much **harder** than Task-IL
Catastrophic Forgetting (CF)

- CF is a **general** challenge in CGL
- **Old** knowledge covered by **new** ones.
Contents

• Continual Graph Learning
• Related Work
• CaT
• PUMA
Related Work

• Regularisation-based: TWP

\[ L_1 (\text{Eq. 9}) \xrightarrow{\text{Eq. 3}} I_1^{\text{loss}} \]

Topological Structure Preserving

\[ e_1^{(l)} = [e_{11}^{(l)}, e_{12}^{(l)}, e_{13}^{(l)}] \]

\[ e_1^{(l)}, e_2^{(l)}, \ldots, [e_{d_1}^{(l)}] \xrightarrow{\text{Eq. 6}} I_1^{s} \]

\[ I_1 = \lambda_1 I_1^{\text{loss}} + \lambda_2 I_1^{s} (\text{Eq. 7}) \]

• Architecture-based: HPNs
Related Work

• Replay-based
Related Work

• Replay-based: ER-GNN
  – Mean feature
  – Coverage Maximization
  – Influence Maximization

• SSM
Contents

• Continual Graph Learning
• Related Work
• CaT
• PUMA
CaT 🐱: Balanced Continual Graph Learning with Graph Condensation

Yilun Liu, Ruihong Qiu, Zi Huang
The University of Queensland
yilun.liu@uq.edu.au

https://github.com/superallen13/CaT-CGL
Batch Training in Replay-based CGL

• Sample and combine
Challenges in Replay

• **Large** storage requirement

• **Imbalanced** training graph size
Condense and Train (CaT) Framework

(a) Typical Replay-based CGL Framework

(b) CaT

Historical Tasks

Future Tasks
Graph Condensation

- Smaller yet effective
- Model **performance** trained on **condensed** graph matches on **original** graph
- **Bi-level** optimisation problem

\[
\min_{\tilde{G}} \mathcal{L}(G; \tilde{\theta}), \text{s.t.} \tilde{\theta} = \arg\min_{\theta} \mathcal{L}(\tilde{G}; \theta)
\]
Condensed Graph Memory (CGM)

- **Distribution matching** method.
- **Random** GNN encoder to obtain latent features.
- Minimise **MMD** losses

\[
\ell_{\text{MMD}} = \sum_{c \in C_k} r_c \cdot \| \text{Mean}(E_{k,c}) - \text{Mean}(\tilde{E}_{k,c}) \|^2
\]
Train in Memory (TiM)

- Condense **incoming** graph
- **Balanced** replayed graphs.

---

**Algorithm 2:** Overall procedure of CaT

**Input:** A streaming of tasks \( \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\} \)

**Output:** \( \text{GNN}_K \)

1. Initialise a CGL model \( \text{GNN}_0 \);
2. Initialise an empty memory bank \( \mathcal{M}_0 \);
3. for \( k \leftarrow 1 \) to \( K \) do
   4. Extract incoming graph \( \mathcal{G}_k \) from \( \mathcal{T}_k \);
   5. Obtain \( \tilde{\mathcal{G}}_k \) by CGM; /* Algorithm 1 */
   6. \( \mathcal{M}_k = \mathcal{M}_{k-1} \cup \tilde{\mathcal{G}}_k \); /* Eq. 11 */
   7. Update \( \text{GNN}_{k-1} \) to \( \text{GNN}_k \); /* Eq. 12 */
4 end
Implementation

• Baselines
  • Joint: full-size.
  • ER-GNN: informative nodes.
  • SSM: subgraphs.

• Metrics
  • Average performance (AP): \( \frac{1}{k} \sum_{i=1}^{k} m_{k,i} \)
  • Backward transfer (BWT):
    \( \frac{1}{k-1} \sum_{i=1}^{k-1} m_{k,i} - m_{i,i} \)

• Experiment settings
  • Dataset splitting [CGLB, NeurIPS 2022]
    – Each task contains two classes.
  • Task incremental learning (task-IL)
    – Classification heads are growing.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
<th>Features</th>
<th>Classes</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoraFull</td>
<td>19,793</td>
<td>130,622</td>
<td>8,710</td>
<td>70</td>
<td>35</td>
</tr>
<tr>
<td>Arxiv</td>
<td>169,343</td>
<td>1,166,243</td>
<td>128</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Reddit</td>
<td>227,853</td>
<td>114,615,892</td>
<td>602</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Products</td>
<td>2,449,028</td>
<td>61,859,036</td>
<td>100</td>
<td>46</td>
<td>23</td>
</tr>
</tbody>
</table>
**Overall Results**

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
<th>CoraFull</th>
<th>Arxiv</th>
<th>Reddit</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AP (%) ↑</td>
<td>BWT (%) ↑</td>
<td>AP (%) ↑</td>
<td>BWT (%) ↑</td>
</tr>
<tr>
<td>Lower bound</td>
<td>Finetuning</td>
<td>2.2±0.0</td>
<td>-96.6±0.1</td>
<td>5.0±0.0</td>
<td>-96.7±0.1</td>
</tr>
<tr>
<td>Regularisation</td>
<td>EWC</td>
<td>2.9±0.2</td>
<td>-96.1±0.3</td>
<td>5.0±0.0</td>
<td>-96.8±0.1</td>
</tr>
<tr>
<td></td>
<td>MAS</td>
<td>2.2±0.0</td>
<td>-94.1±0.6</td>
<td>4.9±0.0</td>
<td>-95.0±0.7</td>
</tr>
<tr>
<td></td>
<td>GEM</td>
<td>2.5±0.1</td>
<td>-96.6±0.1</td>
<td>5.0±0.0</td>
<td>-96.8±0.1</td>
</tr>
<tr>
<td></td>
<td>TWP</td>
<td>21.2±3.2</td>
<td>-67.4±1.6</td>
<td>4.3±1.1</td>
<td>-93.0±8.3</td>
</tr>
<tr>
<td>Distillation</td>
<td>LWF</td>
<td>2.2±0.0</td>
<td>-96.6±0.1</td>
<td>5.0±0.0</td>
<td>-96.8±0.1</td>
</tr>
<tr>
<td>Replay</td>
<td>ER-GNN</td>
<td>4.0±0.7</td>
<td>-94.3±0.9</td>
<td>30.8±0.6</td>
<td>-68.3±0.7</td>
</tr>
<tr>
<td></td>
<td>SSM</td>
<td>16.2±2.8</td>
<td>-82.1±2.9</td>
<td>35.1±1.8</td>
<td>-63.7±1.9</td>
</tr>
<tr>
<td>Full dataset</td>
<td>Joint</td>
<td>85.3±0.1</td>
<td>-2.7±0.0</td>
<td>63.5±0.3</td>
<td>-15.7±0.4</td>
</tr>
<tr>
<td>Ours</td>
<td>CaT</td>
<td><strong>64.5±1.4</strong></td>
<td><strong>-3.3±2.6</strong></td>
<td><strong>66.0±1.1</strong></td>
<td><strong>-13.1±1.0</strong></td>
</tr>
</tbody>
</table>

- **Replayed**-based methods are overall **better**.
- **CaT** is the **best**, sometimes can **match** the ideal **Joint** scenario.
Effectiveness of CGM

- Use TiM for ER-GNN and SSM
- **CGM is more effective** than other memory banks
Visualisation of CGM

- Good coverage of the original distribution
TiM for Catastrophic Forgetting

- TiM has a **balanced** learning to solve CF.
Conclusion

1. **CGM**: Graph condensation gives a small yet effective memory bank

2. **TiM**: A training scheme for balanced continual learning

https://github.com/superallen13/CaT-CGL
Contents

• Continual Graph Learning
• Related Work
• CaT
• PUMA
PUMA 📒: Efficient Continual Graph Learning via Retraining with Pseudo-label Guided Graph Condensation

Yilun Liu, Ruihong Qiu, Yanran Tang, Zi Huang
The University of Queensland
yilun.liu@uq.edu.au

https://github.com/superallen13/PUMA
Problems of CaT

1. **Unlabelled** nodes in streaming graph data

2. **Still imbalance** historical knowledge

---

![Graph showing accuracy and loss over epochs for Task 0 and Task 1, with Task 0 Retrain Accuracy and Task 1 Retrain Accuracy also included.](image-url)
Problems of CaT

3. **Slow** condensation and training
PUMA Framework
Pseudo Labelling-Guided CGM

• An extra classifier for pseudo labels

• Select unlabelled nodes with a high confidence score

• Condense both labelled and confidently pseudo labelled nodes
Train from Scratch

- **No** more *continual* training, but retraining *from scratch*

- **Balanced** historical knowledge and incoming knowledge.
Fast Condensation and Training

• One-time propagation

• Wide graph encoders

• They are fast in calculation and not sacrificing performance. Details in paper
### Overall Performance

<table>
<thead>
<tr>
<th>Category</th>
<th>Methods</th>
<th>CoraFull</th>
<th>Arxiv</th>
<th>Reddit</th>
<th>Products</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AP (%) ↑</td>
<td>BWT (%) ↑</td>
<td>AP (%) ↑</td>
<td>BWT (%) ↑</td>
</tr>
<tr>
<td>Lower bound</td>
<td>Finetuning</td>
<td>2.2±0.0</td>
<td>-96.6±0.1</td>
<td>5.0±0.0</td>
<td>-96.7±0.1</td>
</tr>
<tr>
<td>Regularisation</td>
<td>EWC</td>
<td>2.9±0.2</td>
<td>-96.1±0.3</td>
<td>5.0±0.0</td>
<td>-96.8±0.1</td>
</tr>
<tr>
<td></td>
<td>MAS</td>
<td>2.2±0.0</td>
<td>-94.1±0.6</td>
<td>4.9±0.0</td>
<td>-95.0±0.7</td>
</tr>
<tr>
<td></td>
<td>GEM</td>
<td>2.5±0.1</td>
<td>-96.6±0.1</td>
<td>5.0±0.0</td>
<td>-96.8±0.1</td>
</tr>
<tr>
<td></td>
<td>TWP</td>
<td>21.2±3.2</td>
<td>-67.4±1.6</td>
<td>4.3±1.1</td>
<td>-93.0±8.3</td>
</tr>
<tr>
<td>Distillation</td>
<td>LWF</td>
<td>2.2±0.0</td>
<td>-96.6±0.1</td>
<td>5.0±0.0</td>
<td>-96.8±0.1</td>
</tr>
<tr>
<td>Replay</td>
<td>ER-GNN</td>
<td>3.1±0.2</td>
<td>-94.6±0.2</td>
<td>23.2±0.5</td>
<td>-77.1±0.5</td>
</tr>
<tr>
<td></td>
<td>SSM</td>
<td>3.5±0.5</td>
<td>-94.7±0.5</td>
<td>26.4±0.8</td>
<td>-73.7±0.9</td>
</tr>
<tr>
<td>Full dataset</td>
<td>Joint</td>
<td>85.3±0.1</td>
<td>-2.7±0.0</td>
<td>63.5±0.3</td>
<td>-15.7±0.4</td>
</tr>
<tr>
<td>Condensation</td>
<td>CaT (ours)</td>
<td>68.5±0.9</td>
<td>-5.7±1.3</td>
<td>64.9±0.3</td>
<td>-12.5±0.8</td>
</tr>
<tr>
<td></td>
<td>PUMA (ours)</td>
<td>77.9±0.2</td>
<td>-4.2±0.9</td>
<td>67.0±0.1</td>
<td>-12.2±0.3</td>
</tr>
</tbody>
</table>
Effectiveness of Retraining

- Converge higher
Time Efficiency

- PUMA is fast and performs well
Conclusion

1. **Unlabelled** nodes help with condensation
2. **Retraining** improve the performance
3. Careful designs to **accelerate**
Q&A

Ruihong Qiu
The University of Queensland
r.qiu@uq.edu.au