Learning to Propagate for Graph Meta-learning

Lu Liu¹  Tianyi Zhou²  Guodong Long¹  Jing Jiang¹  Chengqi Zhang¹

University of Technology Sydney¹  University of Washington²
Background

Deep Neural Networks

✓

Large Datasets

✗

Deep Neural Networks

✗

Data - Low Stock ...

Data with Privacy/Copyright
Unseen/Rare Classes
Data hard/costly to collect
Background (cont.)

Deep Neural Networks

Large Datasets

Meta-Learning
(Schmidhuber et al. ’87, Bengio et al. ’92)

Data - Low Stock ...

Data with Privacy/Copyright
Unseen/Rare Classes
Data hard/costly to collect
Meta-Learning: Learning how to learn
Meta-Learning: Learn the global knowledge shared by all learners/tasks.

Examples of meta-learners:
- KNN Support Set [Snell et al., 2017]
- Distance Metric [Vinyals et al., 2016]
- Initialization Point [Finn et al., 2017]

Motivation
Graph Meta-Learning: learn to send message between learners/tasks on a graph

Can graph be a meta-learner?

Motivation (cont.)
Graph Meta-Learning: What’s the graph?

e.g. species in the biology taxonomy

Can graph be a meta-learner?
Motivation (cont.)

Graph Meta-Learning:
What’s the graph?

e.g. diseases in the classification coding system

Can graph be a meta-learner?
Motivation (cont.)

Graph Meta-Learning: What’s the graph?

Can graph be a meta-learner?

e.g. merchandise on an e-commerce website

Meta-Learner

Learn

Train

Few Data Available

Home

Products

Shoes

Clothes

Sunglasses

Locations

Seattle

New York

Contact Us

London
Which kind of graph do we use? How does graph relate to meta-learning?

**LEFT**: Visualization of the class prototypes produced by GPN (our model) for few-shot tasks and the associated graph.

**RIGHT**: GPN's propagation mechanism for one step: for each node, its neighbors pass messages (their prototypes) to it according to attention weight $a$, where a gate further choose to accept the message from the neighbors $g+$ or from the class itself $g^*$. 
Problem definition: graph meta-learning

We evaluate graph meta-learning methods on the tasks of few-shot classification.

Traditional few-shot learning setting

- isolated classes
- A graph of organized classes

Our graph meta-learning setting
Gated Propagation Networks

Prototype propagation in GPN:

in each step t+1, each class y aggregates prototypes from its neighbors (parents and children) by multi-head attention, and chooses between the aggregated message or the message from itself by a gate g.
1. **Initialize prototype**: We set the initial prototype for each class $y$ by averaging over all the K-shot samples belonging to class $y$ as in prototypical networks:

$$P_y^0 \triangleq \frac{1}{|\{(x_i, y_i) \in \mathcal{D}^T : y_i = y\}|} \sum_{(x_i, y_i) \in \mathcal{D}^T, y_i = y} f(x_i).$$
1. **Initialize prototype**

2. **Aggregated messages from its neighbors:** At step $t$, for each class $y$, we firstly compute the aggregated messages from its neighbors $N_y$ by a dot-product attention module $a(p, q)$, i.e.,

   $$P^t_{N_y \rightarrow y} \triangleq \sum_{z \in N_y} a(P^t_y, P^t_z) \times P^t_z, \quad a(p, q) = \frac{\langle h_1(p), h_2(q) \rangle}{\|h_1(p)\| \times \|h_2(q)\|}$$
1. Initialize prototype

2. Aggregated messages from its neighbors

3. **Apply a gate:** We apply a gate $g$ to make decisions of whether accepting messages from its neighbors or message from itself, i.e.

$$P_{y}^{t+1} \triangleq gP_{y}^{t+1} + (1 - g)P_{N_{y} \rightarrow y}^{t+1},$$

$$g = \frac{\exp[\gamma \cos(P_{y}^{0}, P_{y}^{t+1})]}{\exp[\gamma \cos(P_{y}^{0}, P_{y}^{t+1})] + \exp[\gamma \cos(P_{y}^{0}, P_{N_{y} \rightarrow y}^{t+1})]}.$$
1. Initialize prototype

2. Aggregated messages from its neighbors

3. Apply a gate

Note:
To capture different types of relation and jointly use them for propagation, we combine the results of $k$ attentive and gated propagation modules with untied parameters.

$$P_{y}^{t+1} = \frac{1}{k} \sum_{i=1}^{k} P_{y}^{t+1}[i]$$
Gated Propagation Networks (Cont.)

1. Initialize prototype

2. Aggregated messages from its neighbors

3. Apply a gate

4. The final prototype is given as the weighted sum of the initial prototype and the refined prototype:

\[ P_y \triangleq \lambda \times P_y^0 + (1 - \lambda) \times P_y^T \]
- **Generating training tasks** by subgraph sampling: random sampling and snowball sampling. Random sampling captures strongly-related classes. Snowball sampling captures weakly-related classes.

- **Building propagation pathways** by maximum spanning tree. Only propagate through the most related / close classes according to cosine similarity.

- **Curriculum learning**
  Early stage: train on traditional supervised learning tasks
  Later stage: train on few-shot learning tasks
Algorithm 1 GPN Training

\textbf{Input:} $G = (\mathcal{Y}, E)$, memory update interval $m$, propagation steps $\mathcal{T}$, total episodes $\tau_{total}$;

1: \textbf{Initialization:} $\Theta^{cnn}, \Theta^{prop}, \Theta^{fc}, \tau \leftarrow 0$;
2: \textbf{for} $\tau \in \{1, \cdots, \tau_{total}\}$ \textbf{do}
3: \quad \textbf{if} $\tau \mod m = 0$ \textbf{then}
4: \quad \quad Update prototypes in memory by Eq. (3);
5: \quad \textbf{end if}
6: \quad Draw $\alpha \sim \text{Unif}[0, 1]$;
7: \quad \textbf{if} $\alpha < 0.9^{20\tau/\tau_{t}}$ \textbf{then}
8: \quad \quad Train a classifier to update $\Theta^{cnn}$ with loss $\sum_{(x,y) \sim \mathcal{D}} - \log \Pr(y|x; \Theta^{cnn}, \Theta^{fc})$;
9: \quad \textbf{else}
10: \quad \quad Sample a few-shot task $T$ as in Sec. 3.3;
11: \quad \quad Construct MST $\mathcal{Y}_{MST}^T$ as in Sec. 3.3;
12: \quad \quad For $y \in \mathcal{Y}_{MST}^T$, compute $P_{y}^0$ by Eq. (3) if $y \in T$, otherwise fetch $P_{y}^0$ from memory;
13: \quad \textbf{for} $t \in \{1, \cdots, \mathcal{T}\}$ \textbf{do}
14: \quad \quad For all $y \in \mathcal{Y}_{MST}^T$, concurrently update their prototypes $P_{y}^t$ by Eq. (4)-(6);
15: \quad \textbf{end for}
16: \quad Compute $P_{y}$ for $y \in \mathcal{Y}_{MST}^T$ by Eq.(7);
17: \quad Compute $\log \Pr(y|x; \Theta^{cnn}, \Theta^{prop})$ by Eq. (2) for all samples $(x, y)$ in task $T$;
18: \quad Update $\Theta^{cnn}$ and $\Theta^{prop}$ by minimizing $\sum_{(x,y) \sim \mathcal{D}^T} - \log \Pr(y|x; \Theta^{cnn}, \Theta^{prop})$;
19: \quad \textbf{end if}
20: \textbf{end for}
Experimental Results

Table 3: Validation accuracy (mean±CI%95) on 600 test tasks achieved by GPN and baselines on tieredImageNet-Close with few-shot tasks generated by random sampling.

<table>
<thead>
<tr>
<th>Model</th>
<th>5way1shot</th>
<th>5way5shot</th>
<th>10way1shot</th>
<th>10way5shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototypical Net [23]</td>
<td>42.87±1.67%</td>
<td>62.68±0.99%</td>
<td>30.65±1.15%</td>
<td>48.64±0.70%</td>
</tr>
<tr>
<td>GNN [6]</td>
<td>42.33±0.80%</td>
<td>59.17±0.69%</td>
<td>30.50±0.57%</td>
<td>44.33±0.72%</td>
</tr>
<tr>
<td>Closer Look [3]</td>
<td>35.07±1.53%</td>
<td>47.48±0.87%</td>
<td>21.58±0.96%</td>
<td>28.01±0.40%</td>
</tr>
<tr>
<td>PPN [15]</td>
<td>41.60±1.59%</td>
<td>63.04±0.97%</td>
<td>28.48±1.09%</td>
<td>48.66±0.70%</td>
</tr>
<tr>
<td>GPN</td>
<td>48.37±1.80%</td>
<td>64.14±1.00%</td>
<td>33.23±1.05%</td>
<td>50.50±0.70%</td>
</tr>
<tr>
<td>GPN+</td>
<td>50.54±1.67%</td>
<td>65.74±0.98%</td>
<td>34.74±1.05%</td>
<td>51.50±0.70%</td>
</tr>
</tbody>
</table>

Table 4: Validation accuracy (mean±CI%95) on 600 test tasks achieved by GPN and baselines on tieredImageNet-Close with few-shot tasks generated by snowball sampling.

<table>
<thead>
<tr>
<th>Model</th>
<th>5way1shot</th>
<th>5way5shot</th>
<th>10way1shot</th>
<th>10way5shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototypical Net [23]</td>
<td>35.27±1.63%</td>
<td>52.60±1.17%</td>
<td>26.08±1.04%</td>
<td>41.48±0.76%</td>
</tr>
<tr>
<td>GNN [6]</td>
<td>36.50±1.03%</td>
<td>52.33±0.96%</td>
<td>27.67±1.01%</td>
<td>40.67±0.90%</td>
</tr>
<tr>
<td>Closer Look [3]</td>
<td>34.07±1.63%</td>
<td>47.48±0.87%</td>
<td>21.02±0.99%</td>
<td>33.70±0.44%</td>
</tr>
<tr>
<td>PPN [15]</td>
<td>36.50±1.62%</td>
<td>52.50±1.12%</td>
<td>27.18±1.08%</td>
<td>40.97±0.77%</td>
</tr>
<tr>
<td>GPN</td>
<td>39.56±1.70%</td>
<td>54.35±1.11%</td>
<td>27.99±1.09%</td>
<td>42.50±0.76%</td>
</tr>
<tr>
<td>GPN+</td>
<td>40.78±1.76%</td>
<td>55.47±1.41%</td>
<td>29.46±1.10%</td>
<td>43.76±0.74%</td>
</tr>
</tbody>
</table>
Experimental Results

Table 5: Validation accuracy (mean±CI%95) on 600 test tasks achieved by GPN and baselines on tieredImageNet-Far with few-shot tasks generated by random sampling.

<table>
<thead>
<tr>
<th>Model</th>
<th>5way1shot</th>
<th>5way5shot</th>
<th>10way1shot</th>
<th>10way5shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototypical Net [23]</td>
<td>44.30±1.63%</td>
<td>61.01±1.03%</td>
<td>30.63±1.07%</td>
<td>47.19±0.68%</td>
</tr>
<tr>
<td>GNN [6]</td>
<td>43.67±0.69%</td>
<td>59.33±1.04%</td>
<td>30.17±0.47%</td>
<td>43.00±0.66%</td>
</tr>
<tr>
<td>Closer Look [3]</td>
<td>42.27±1.70%</td>
<td>58.78±0.94%</td>
<td>22.00±0.99%</td>
<td>32.73±0.41%</td>
</tr>
<tr>
<td>PPN [15]</td>
<td>43.63±1.59%</td>
<td>60.20±1.02%</td>
<td>29.55±1.09%</td>
<td>46.72±0.66%</td>
</tr>
<tr>
<td>GPN</td>
<td>47.54±1.68%</td>
<td>64.20±1.01%</td>
<td>31.84±1.10%</td>
<td>48.20±0.69%</td>
</tr>
<tr>
<td>GPN+</td>
<td>47.49±1.67%</td>
<td>64.14±1.02%</td>
<td>31.95±1.15%</td>
<td>48.65±0.66%</td>
</tr>
</tbody>
</table>

Table 6: Validation accuracy (mean±CI%95) on 600 test tasks achieved by GPN and baselines on tieredImageNet-Far with few-shot tasks generated by snowball sampling.

<table>
<thead>
<tr>
<th>Model</th>
<th>5way1shot</th>
<th>5way5shot</th>
<th>10way1shot</th>
<th>10way5shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prototypical Net [23]</td>
<td>44.57±1.67%</td>
<td>62.35±1.06%</td>
<td>29.88±1.11%</td>
<td>46.48±0.70%</td>
</tr>
<tr>
<td>GNN [6]</td>
<td>44.00±1.36%</td>
<td>62.00±0.66%</td>
<td>28.50±0.60%</td>
<td>46.17±0.74%</td>
</tr>
<tr>
<td>Closer Look [3]</td>
<td>38.37±1.57%</td>
<td>54.64±0.85%</td>
<td>30.40±1.09%</td>
<td>33.72±0.43%</td>
</tr>
<tr>
<td>PPN [15]</td>
<td>42.40±1.63%</td>
<td>61.37±1.05%</td>
<td>28.67±1.01%</td>
<td>46.02±0.61%</td>
</tr>
<tr>
<td>GPN</td>
<td>47.74±1.76%</td>
<td>63.53±1.03%</td>
<td>32.94±1.16%</td>
<td>47.43±0.67%</td>
</tr>
<tr>
<td>GPN+</td>
<td>47.58±1.70%</td>
<td>63.74±0.95%</td>
<td>32.68±1.17%</td>
<td>47.44±0.71%</td>
</tr>
</tbody>
</table>

5~10 steps/hops between training classes and test classes
Ablation Studies

Table 7: Validation accuracy (mean±CI%95) for possible variants of GPN on tieredImageNet-Close for 5-way-1-shot tasks. Original GPN’s choices are in **bold** fonts. Details of the variants are given in Sec. 4.5.

<table>
<thead>
<tr>
<th>Task Generation</th>
<th>Propagation Mechanism</th>
<th>Training</th>
<th>Model</th>
<th>ACCURACY</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR-S S-S R-S</td>
<td>N→C F→C C→C B→P M→P</td>
<td>AUX MST</td>
<td>M-H M-A A-A</td>
<td>46.20±1.70%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>49.33±1.68%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42.60±1.61%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>37.90±1.50%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>47.90±1.72%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>46.90±1.78%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>41.87±1.72%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45.83±1.64%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>49.40±1.69%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>46.74±1.71%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>50.54±1.67%</strong></td>
</tr>
</tbody>
</table>
Visualization of the class prototypes produced by GPN for few-shot tasks and the associated graph.
Visualization (Cont.)

More …
Prototypes before (top row) and after GPN propagation (bottom row) on tieredImageNet-Close by random sampling for 5-way-1-shot few-shot learning. The prototypes in top row equal to the ones achieved by prototypical network. Different tasks are marked by a different shape, and classes shared by different tasks are highlighted by non-grey colors.

It shows that GPN is capable to map the prototypes of the same class in different tasks to the same region. Comparing to the result of prototypical network, GPN is more powerful in relating different tasks.
Visualization (Cont.)
Takeaways

- **A novel problem**: Graph Meta-learning, which learns to send message between learners/tasks on a graph.

- A meta-learning method based **“Learning to Propagate”**, where the propagation scheme on the graph is a meta-learner.

- **Two new benchmark datasets** for the new problem and thorough **experiments** for comparison, ablation and visualization.