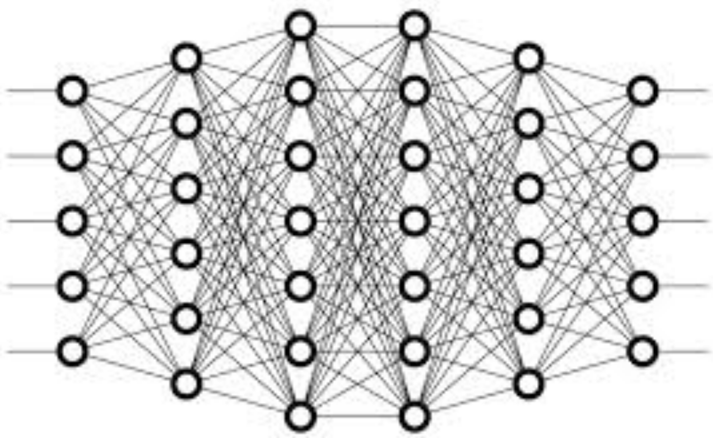
Learning to Propagate for Graph Meta-learning

Lu Liu[1] Tianyi Zhou[2] Guodong Long[1] Jing Jiang[1] Chengqi Zhang[1]

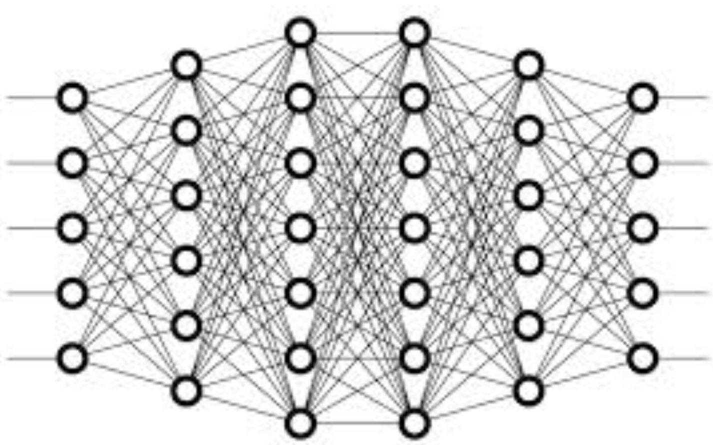
University of Technology Sydney[1] University of Washington[2]

Background

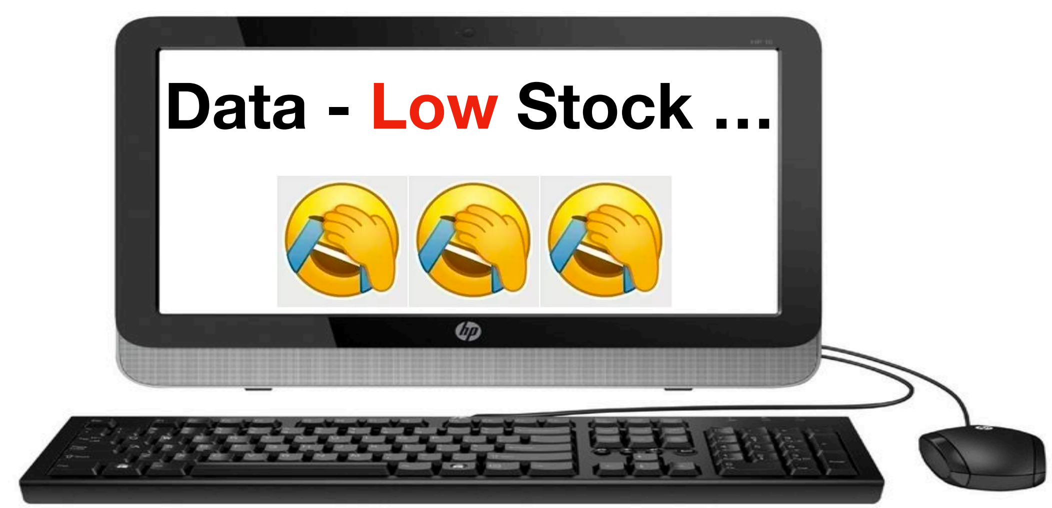


Deep Neural Networks

Large Datasets

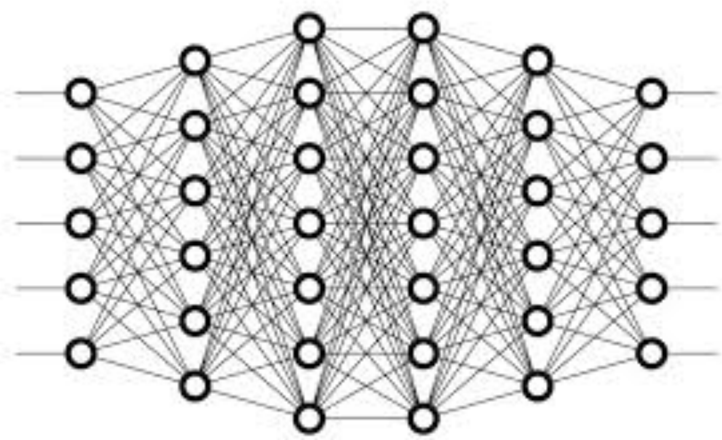


Deep Neural Networks



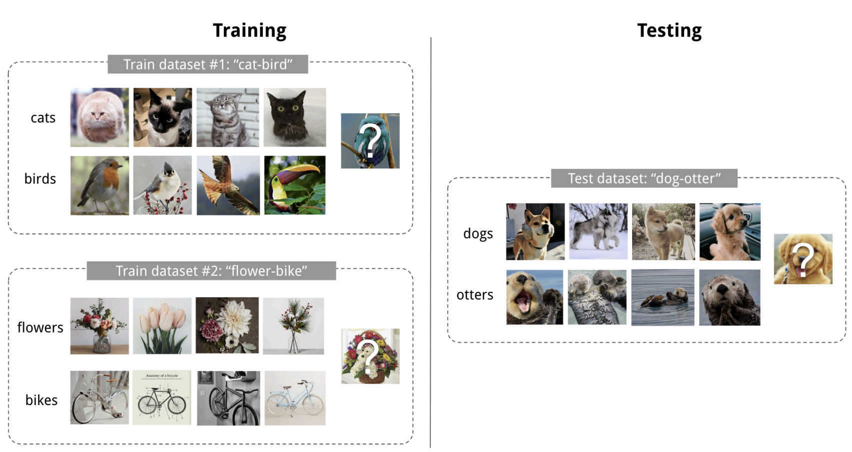
Data with Privacy/Copyright
Unseen/Rare Classes
Data hard/costly to collect

Background (cont.)



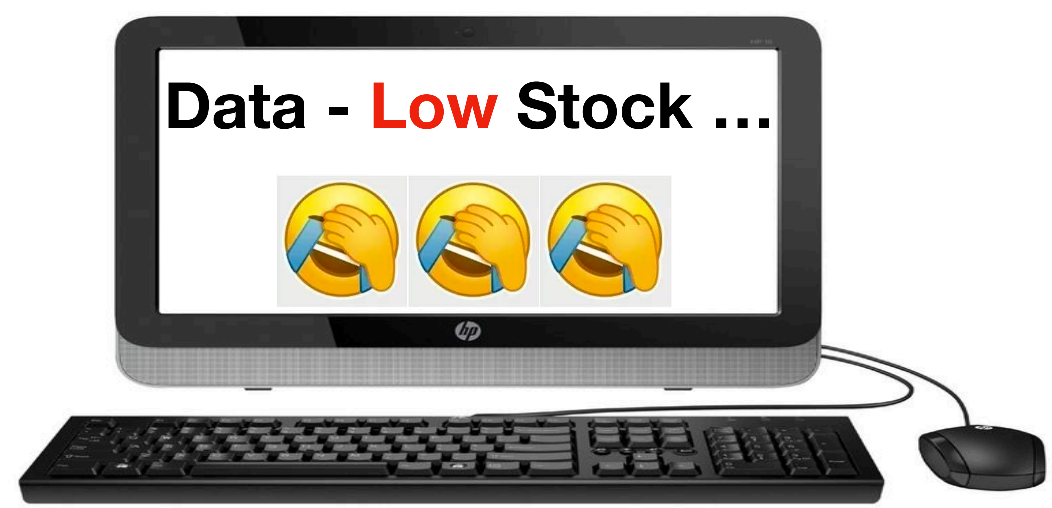
Deep Neural Networks

Large Datasets



Meta-Learning

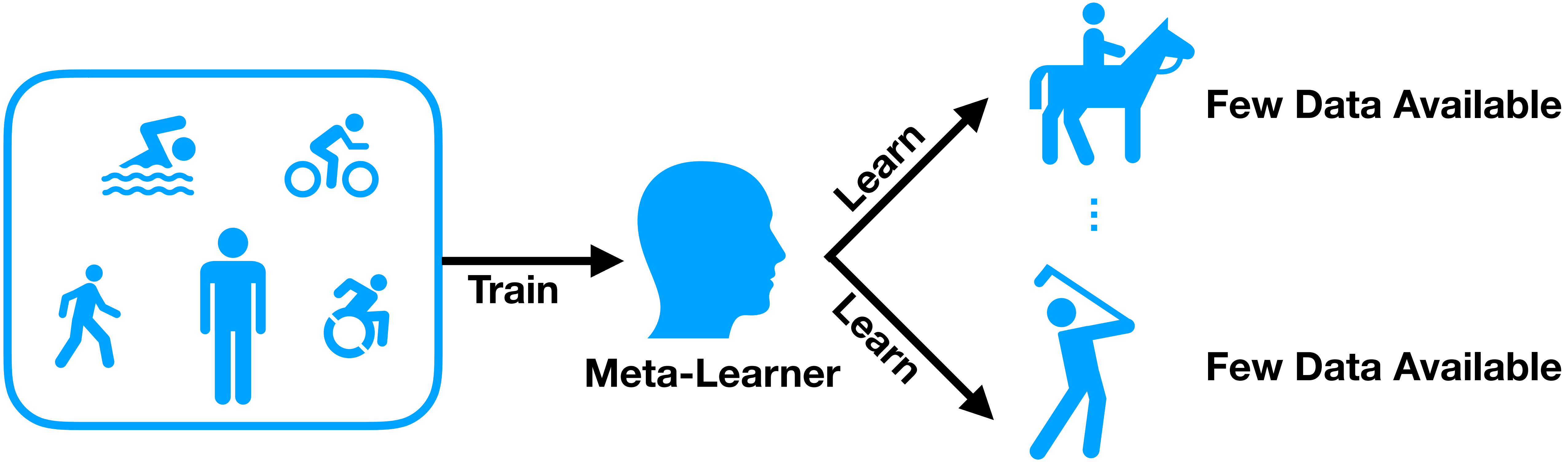
(Schmidhuber et al. '87, Bengio et al. '92)



Data with Privacy/Copyright
Unseen/Rare Classes
Data hard/costly to collect

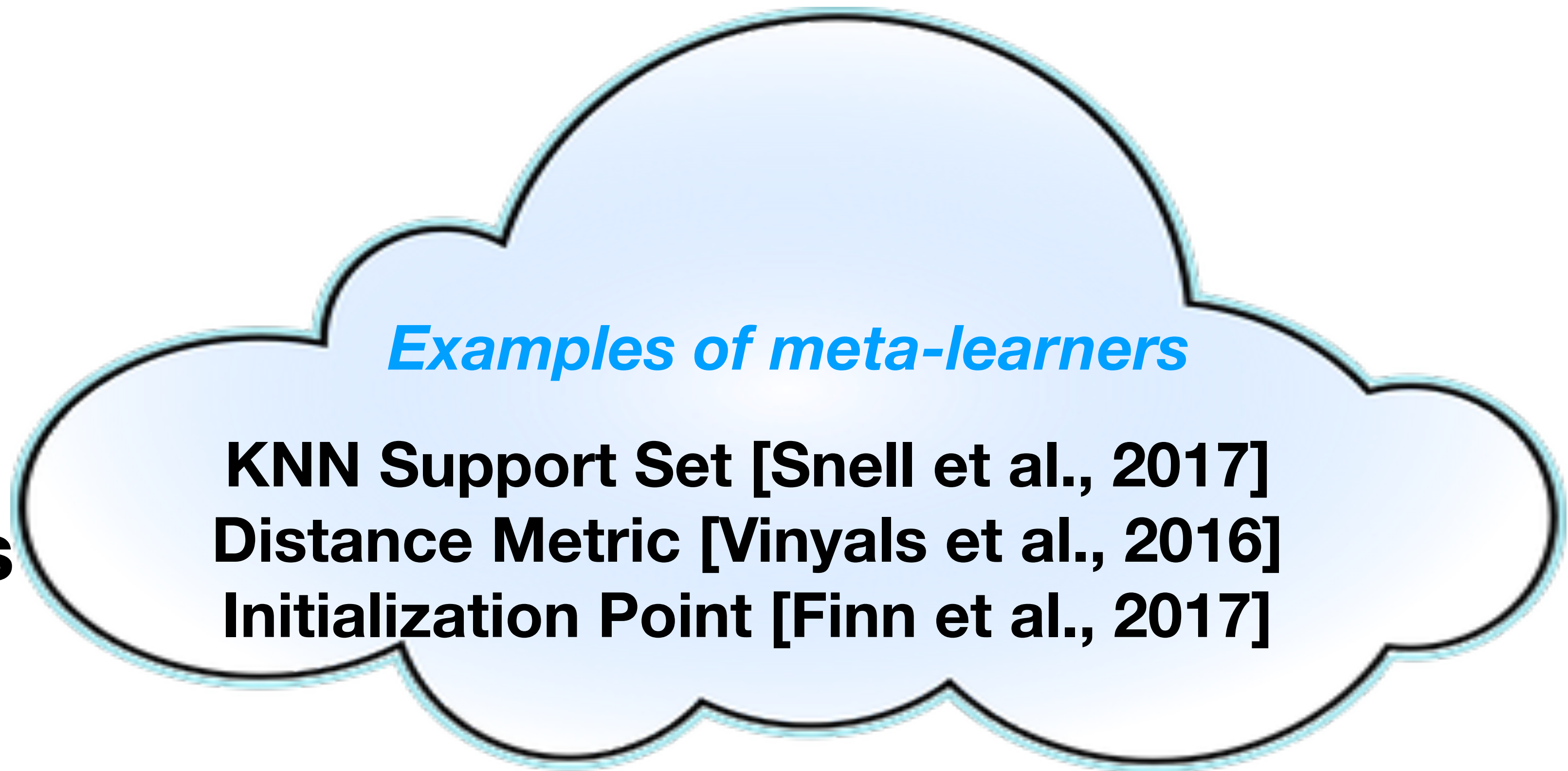
Background (cont.)

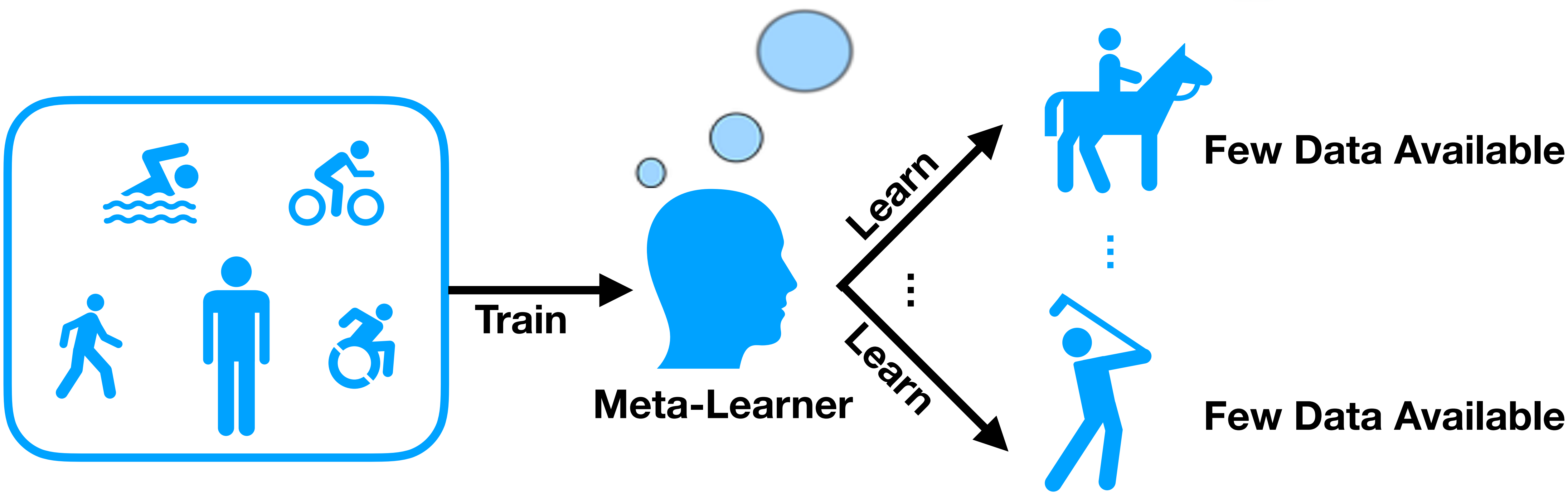
Meta-Learning: Learning how to learn



Motivation

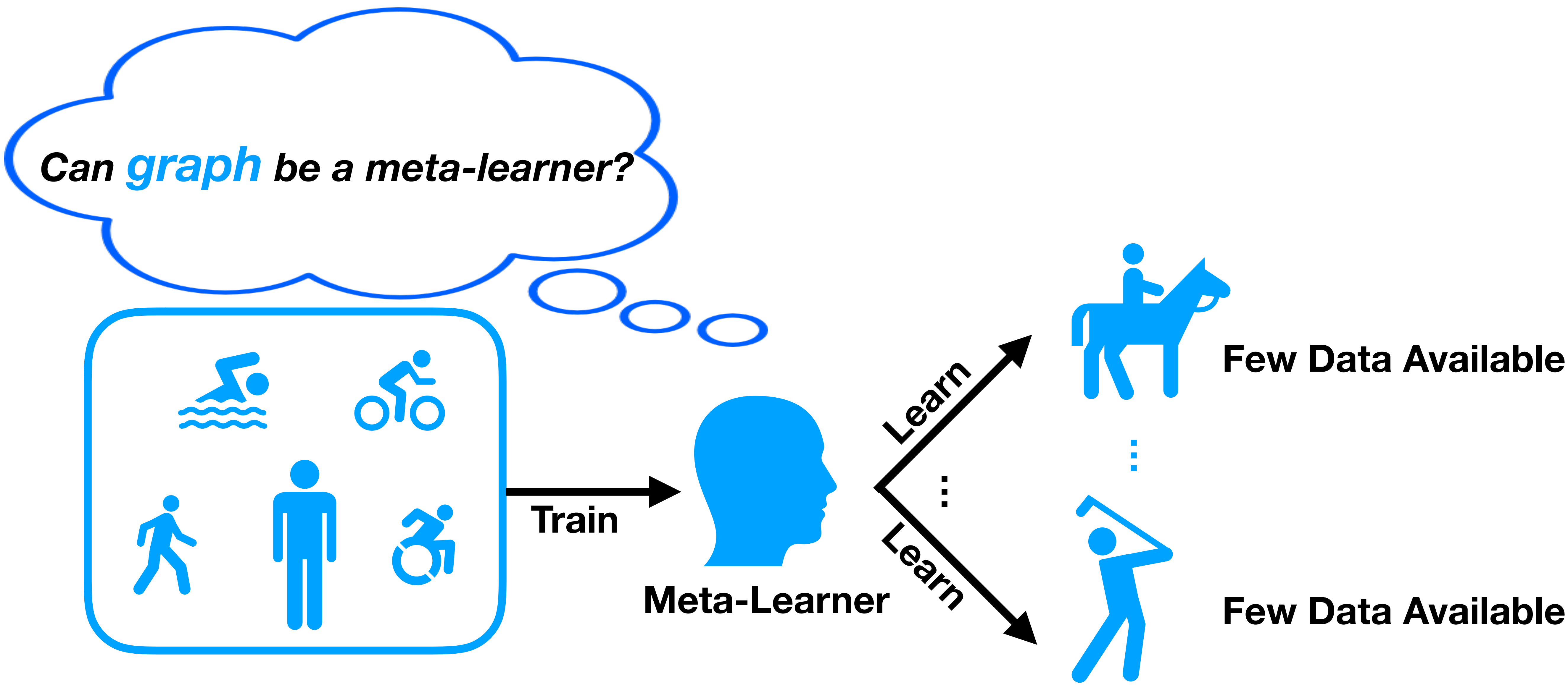
**Meta-Learning:
Learn the global knowledge shared by all learners/tasks**





Motivation (cont.)

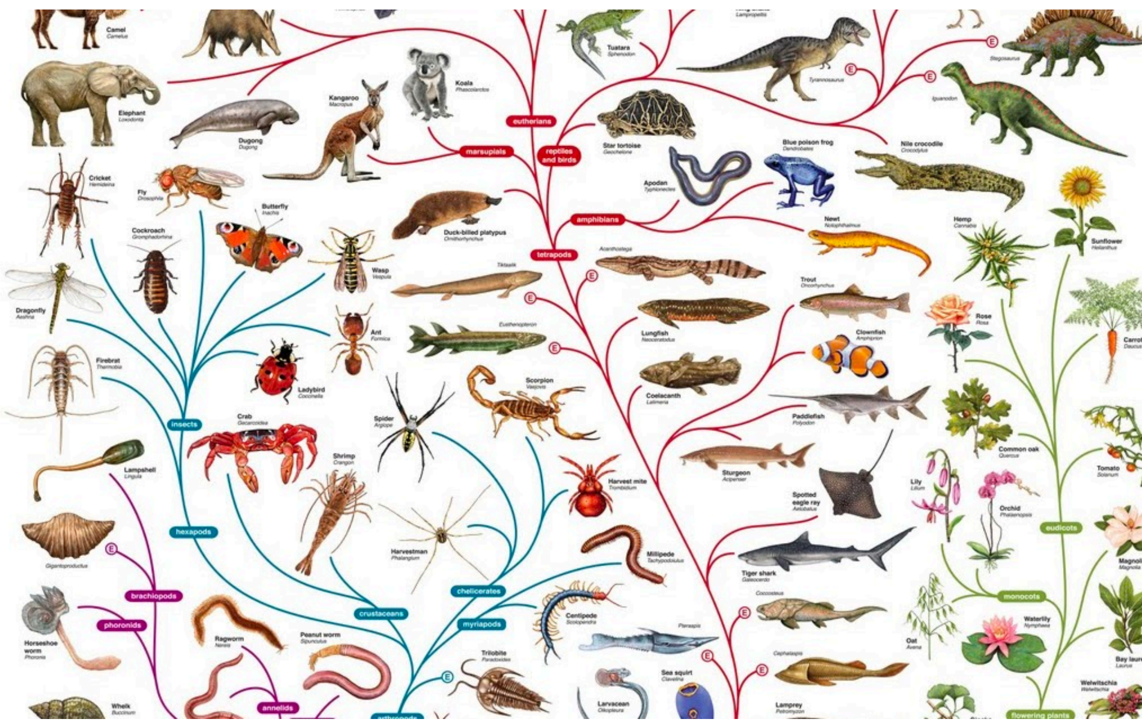
Graph Meta-Learning: learn to send message between learners/tasks on a graph



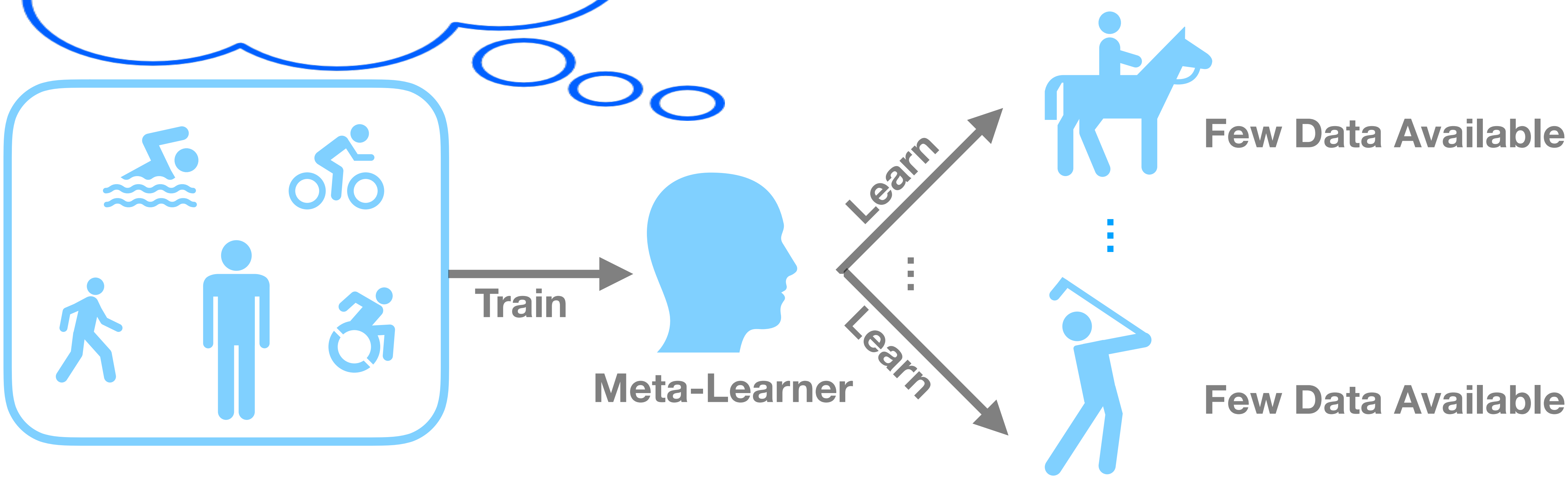
Motivation (cont.)

Graph Meta-Learning:
What's the graph?

e.g. species in the biology taxonomy



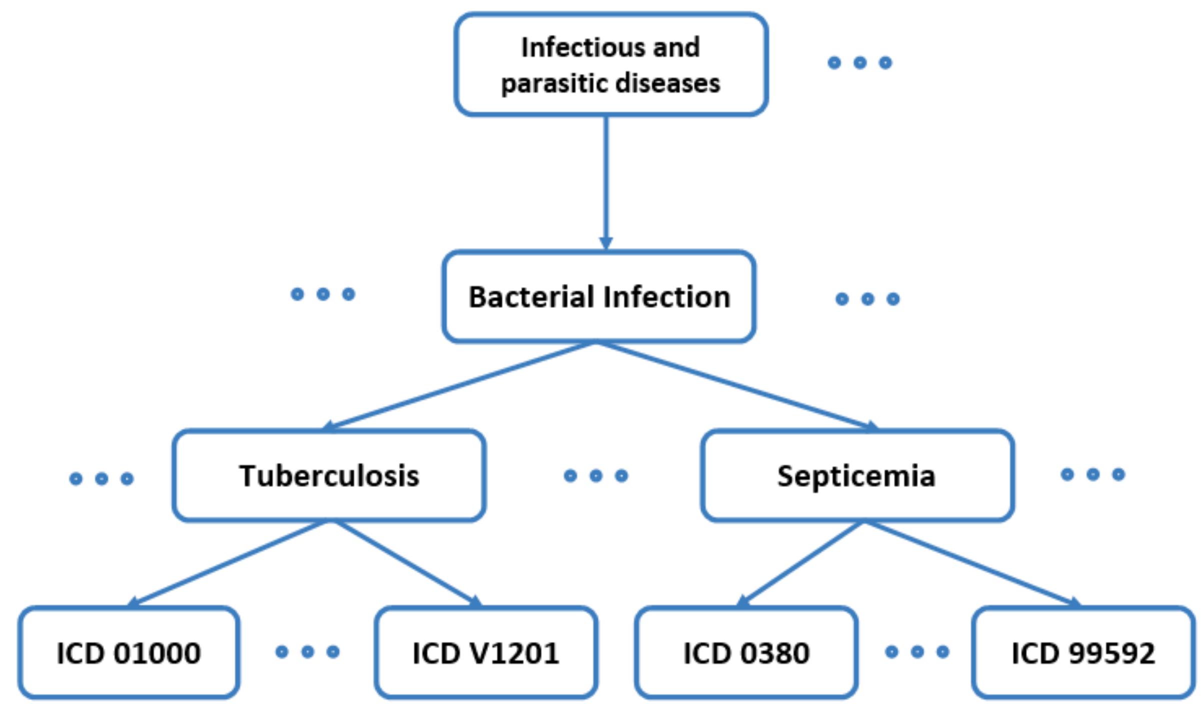
Can graph be a meta-learner?



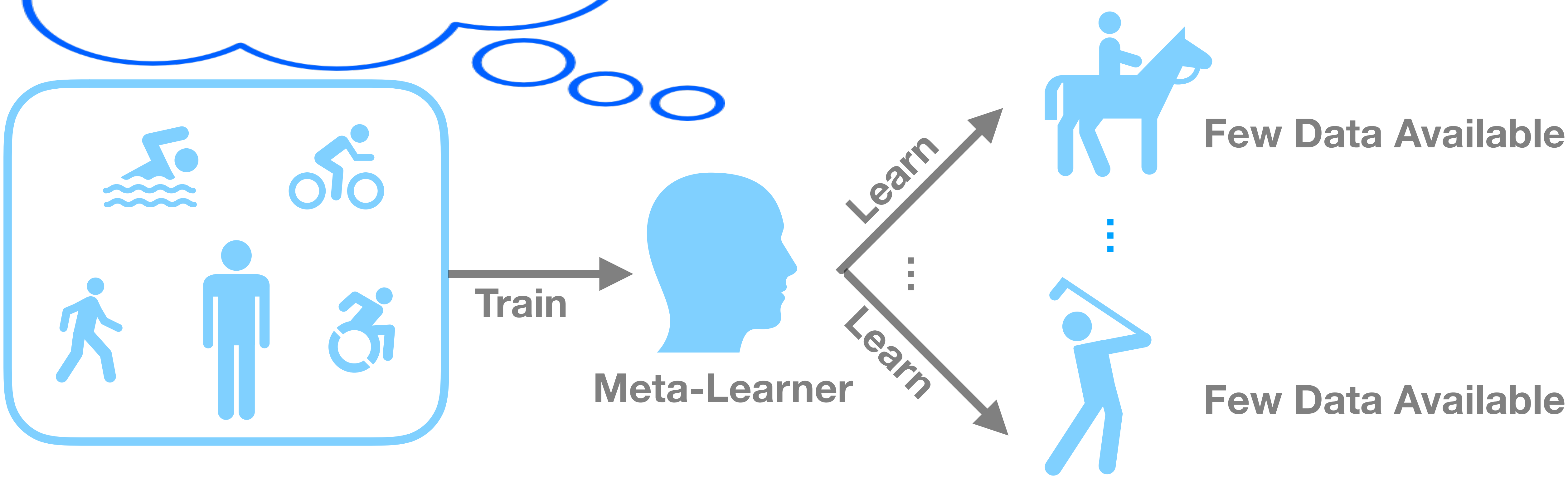
Motivation (cont.)

Graph Meta-Learning:
What's the graph?

e.g. diseases in the classification coding system



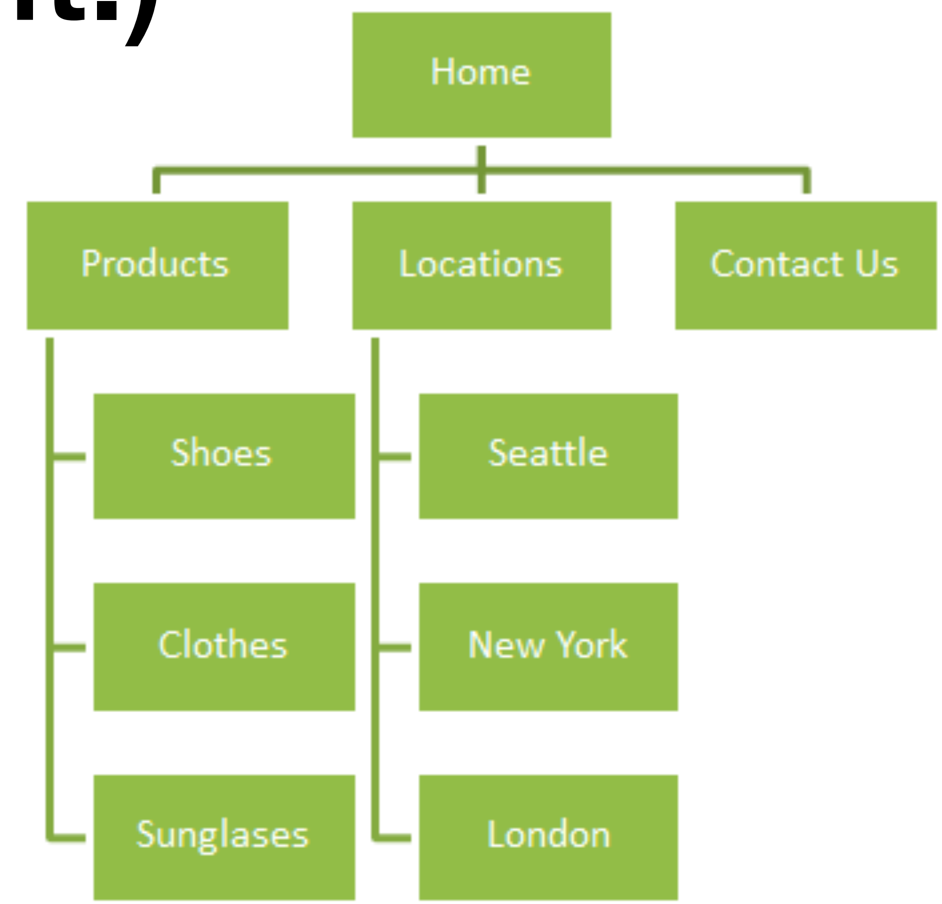
Can graph be a meta-learner?

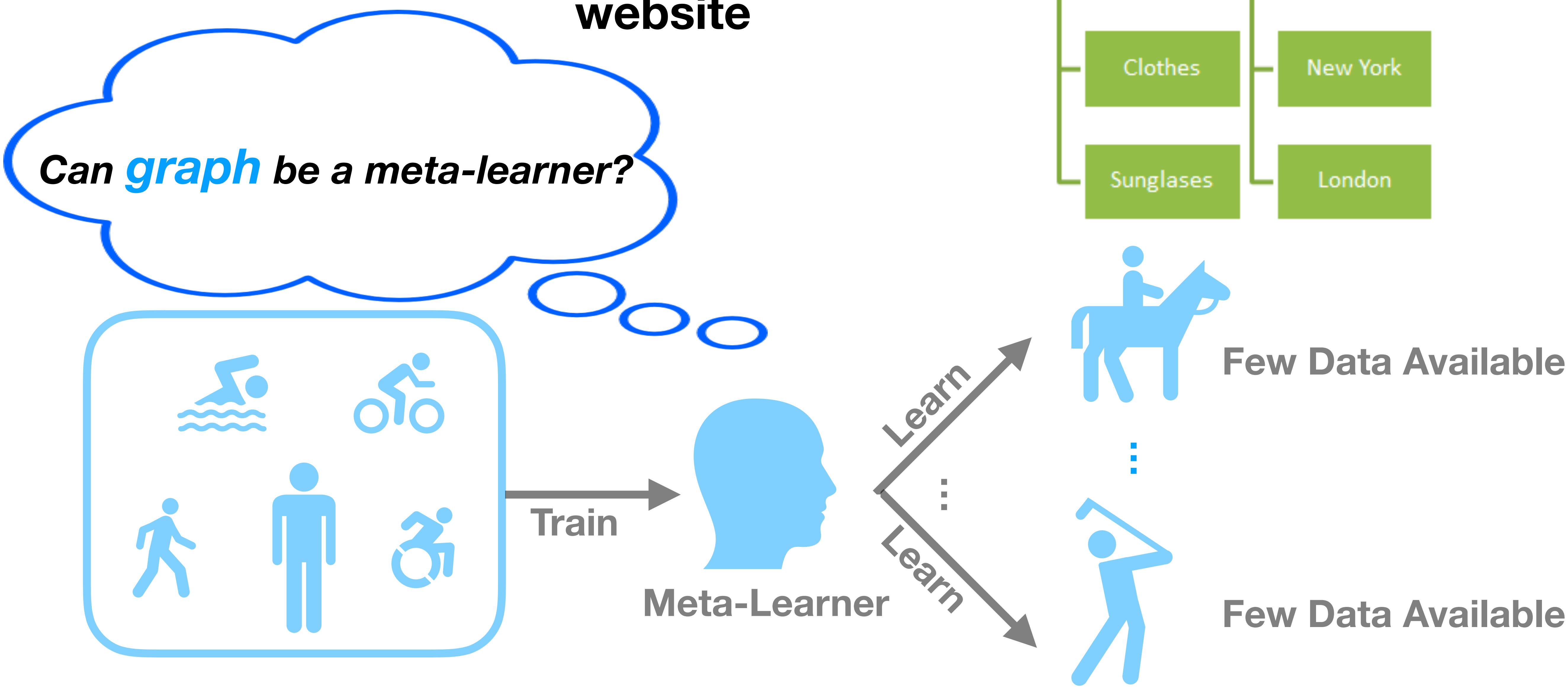


Motivation (cont.)

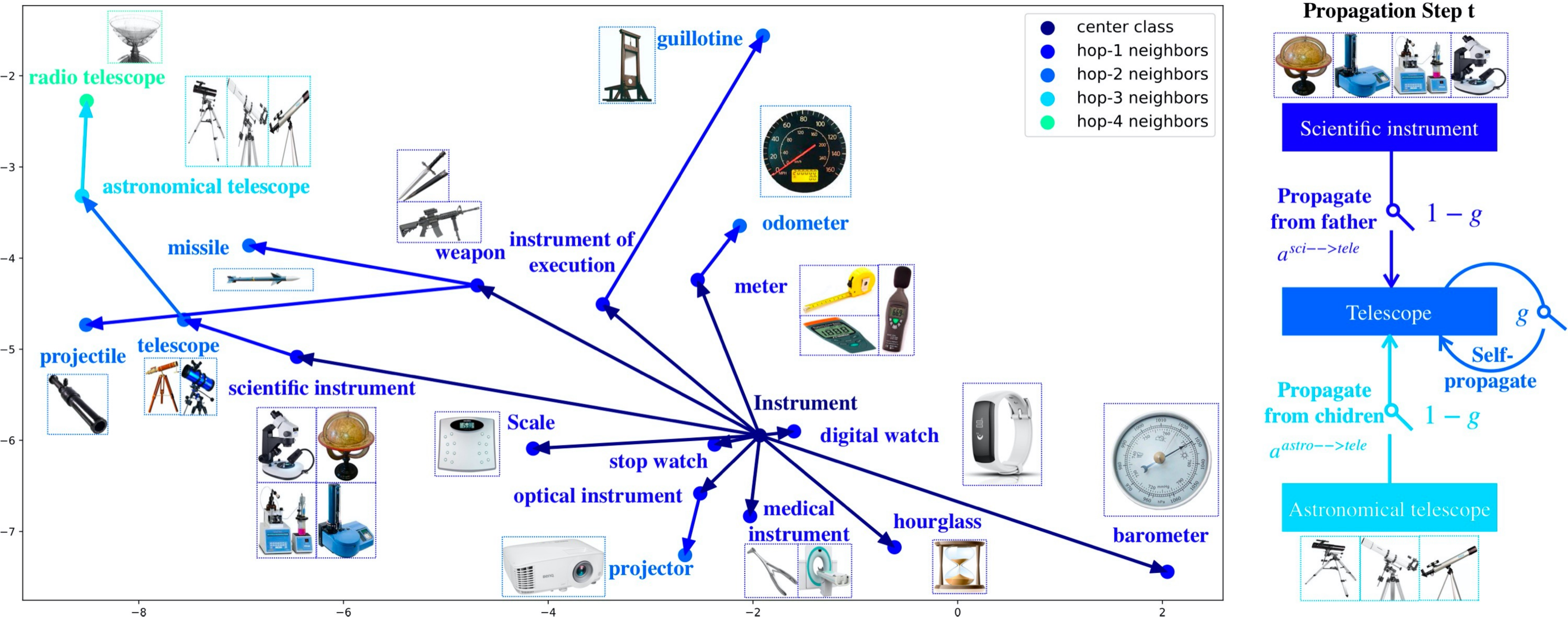
Graph Meta-Learning:
What's the graph?

e.g. merchandise on an e-commerce website





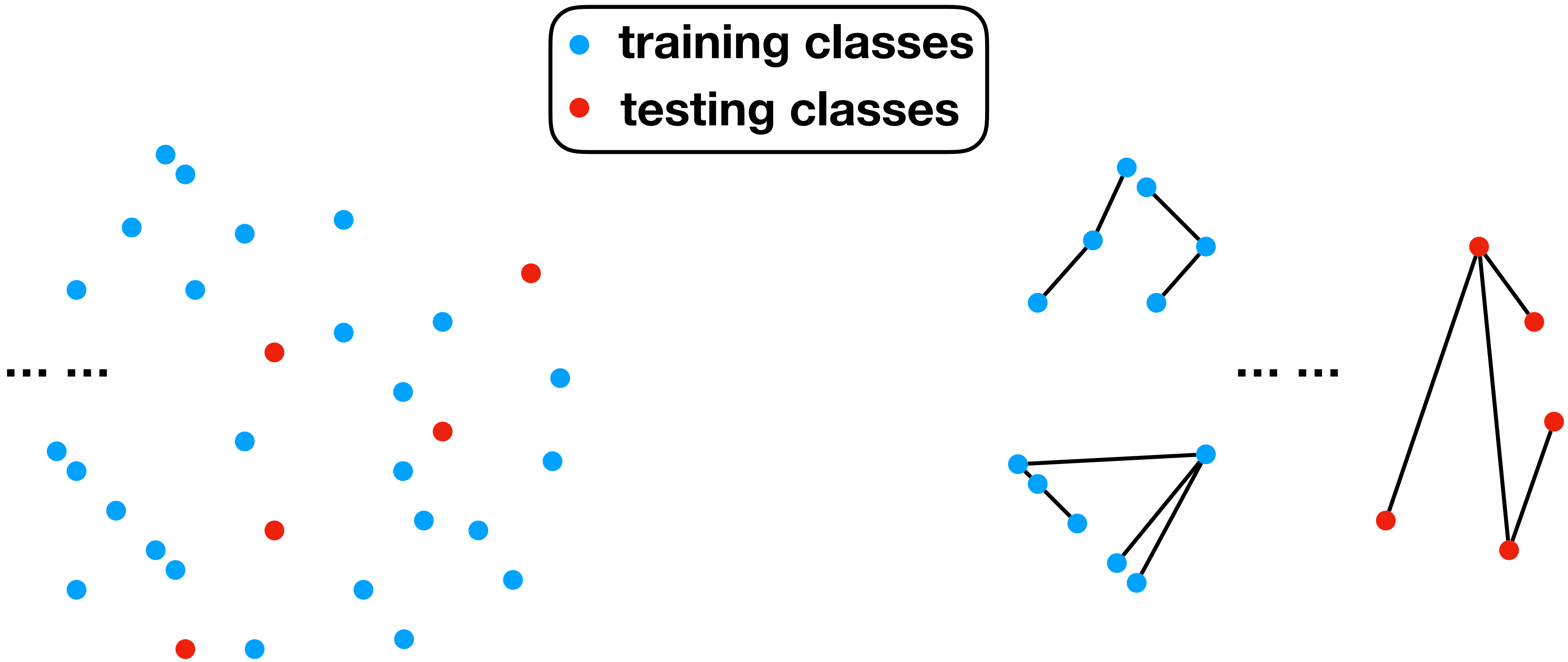
Which kind of graph do we use ?
How does graph relate to meta-learning?



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



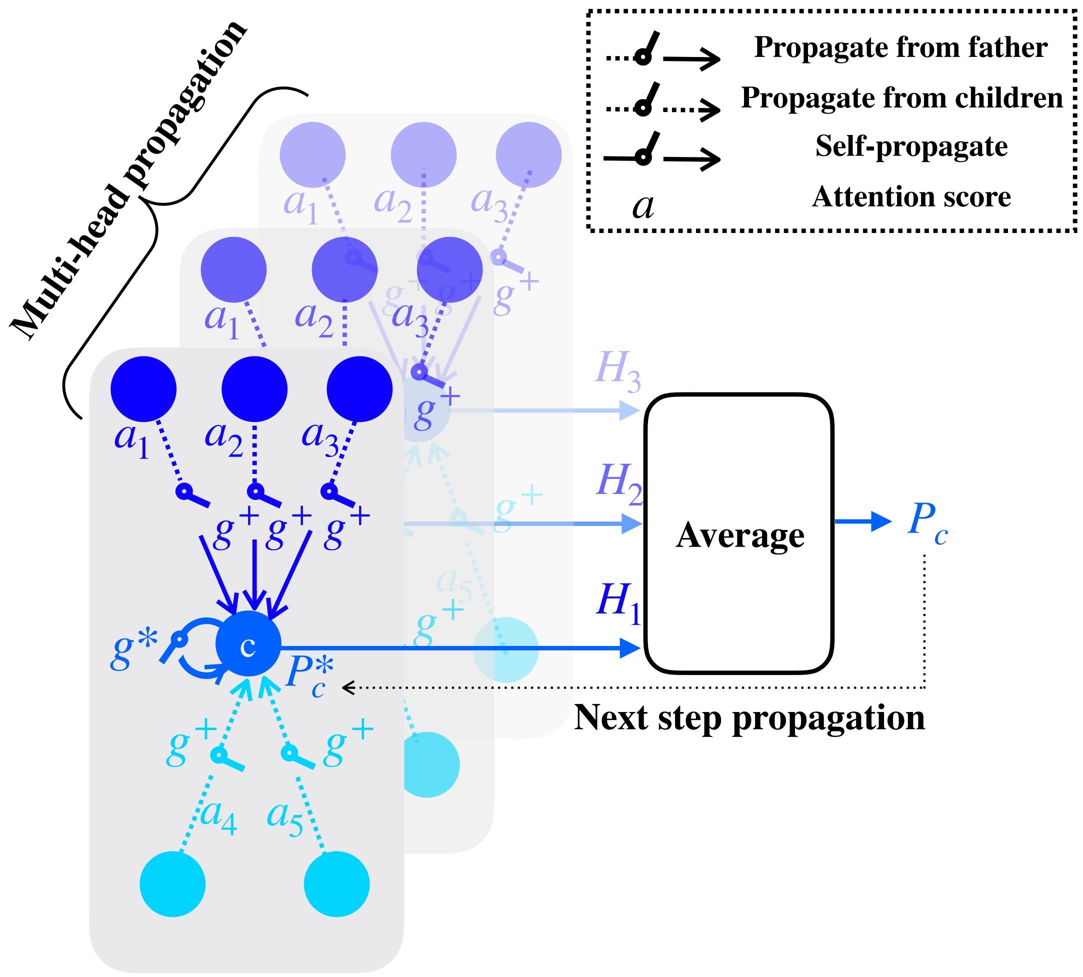
isolated classes

Traditional few-shot learning setting

A graph of organized classes

Our <u>graph</u> meta-learning setting

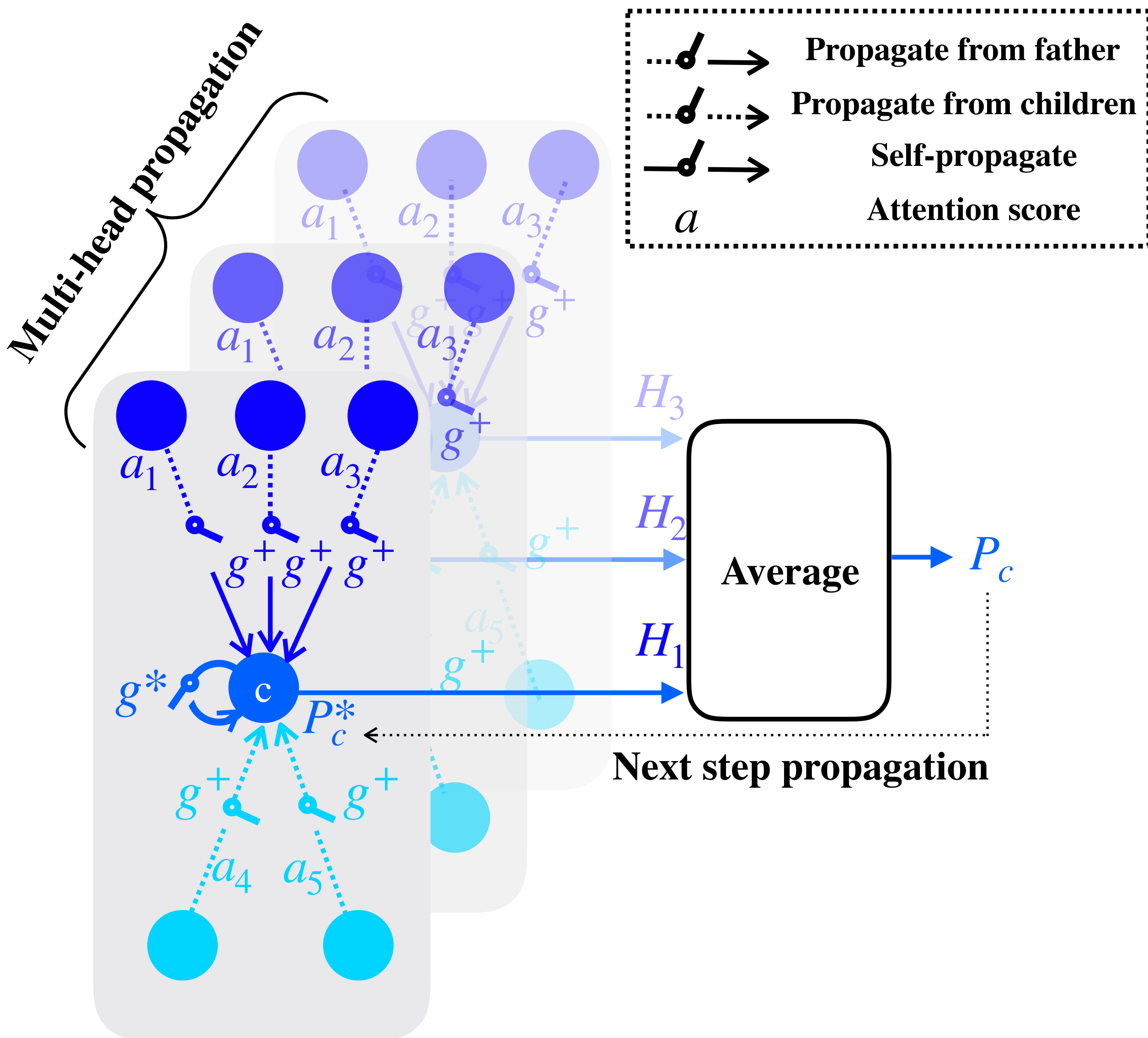
Gated Propagation Networks



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.

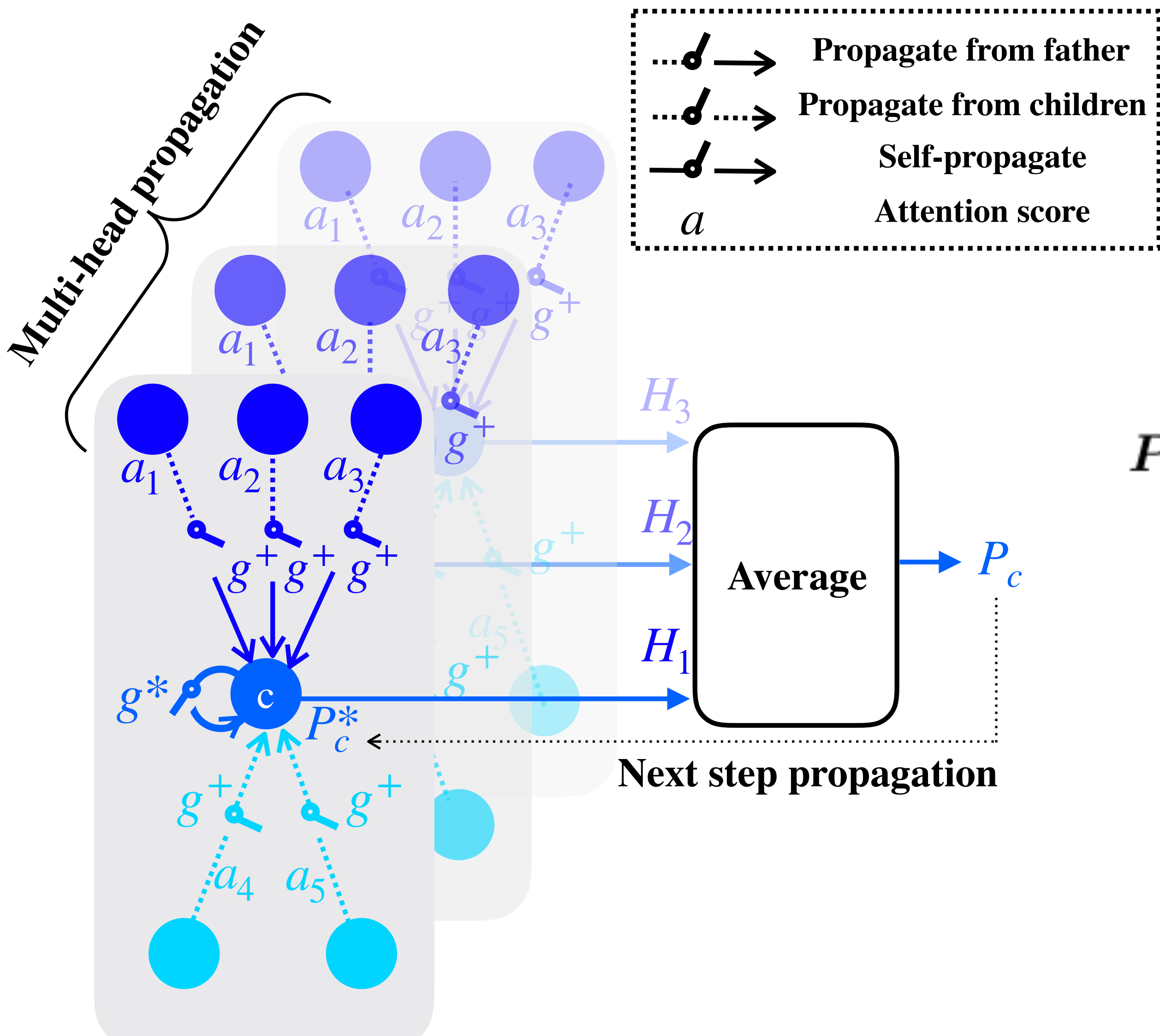
Gated Propagation Networks (Cont.)



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:

$$\boldsymbol{P}_y^0 \triangleq \frac{1}{|\{(\boldsymbol{x}_i, y_i) \in \mathcal{D}^T : y_i = y\}|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}^T, y_i = y} f(\boldsymbol{x}_i).$$

Gated Propagation Networks (Cont.)

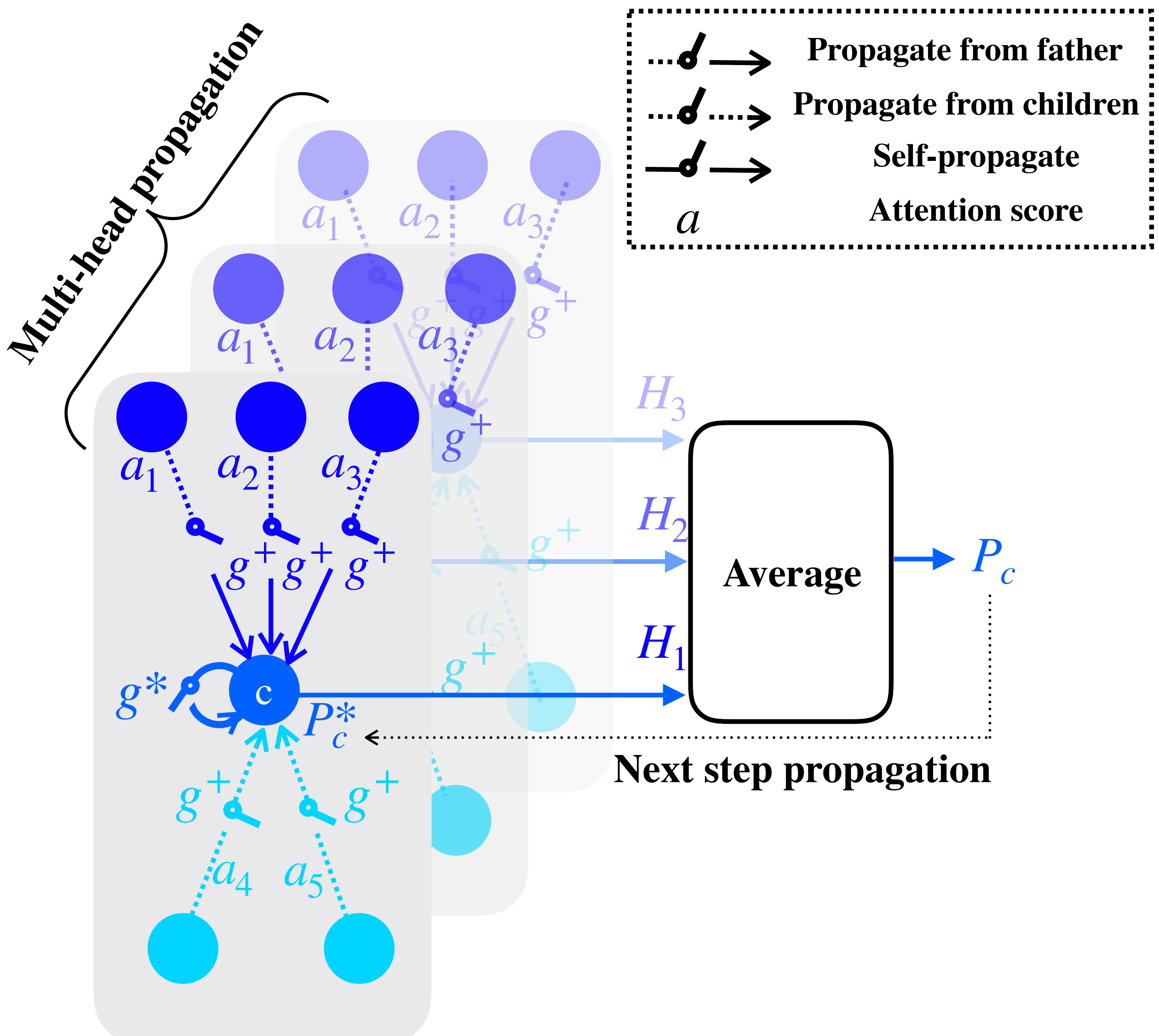


1. **Initialize prototype**

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

$$\boldsymbol{P}^{t+1}_{\mathcal{N}_y \to y} \triangleq \sum_{z \in \mathcal{N}_y} a(\boldsymbol{P}^t_y, \boldsymbol{P}^t_z) \times \boldsymbol{P}^t_z, \quad a(p, q) = \frac{\langle h_1(p), h_2(q) \rangle}{\|h_1(p)\| \times \|h_2(q)\|}$$

Gated Propagation Networks (Cont.)



1. **Initialize prototype**

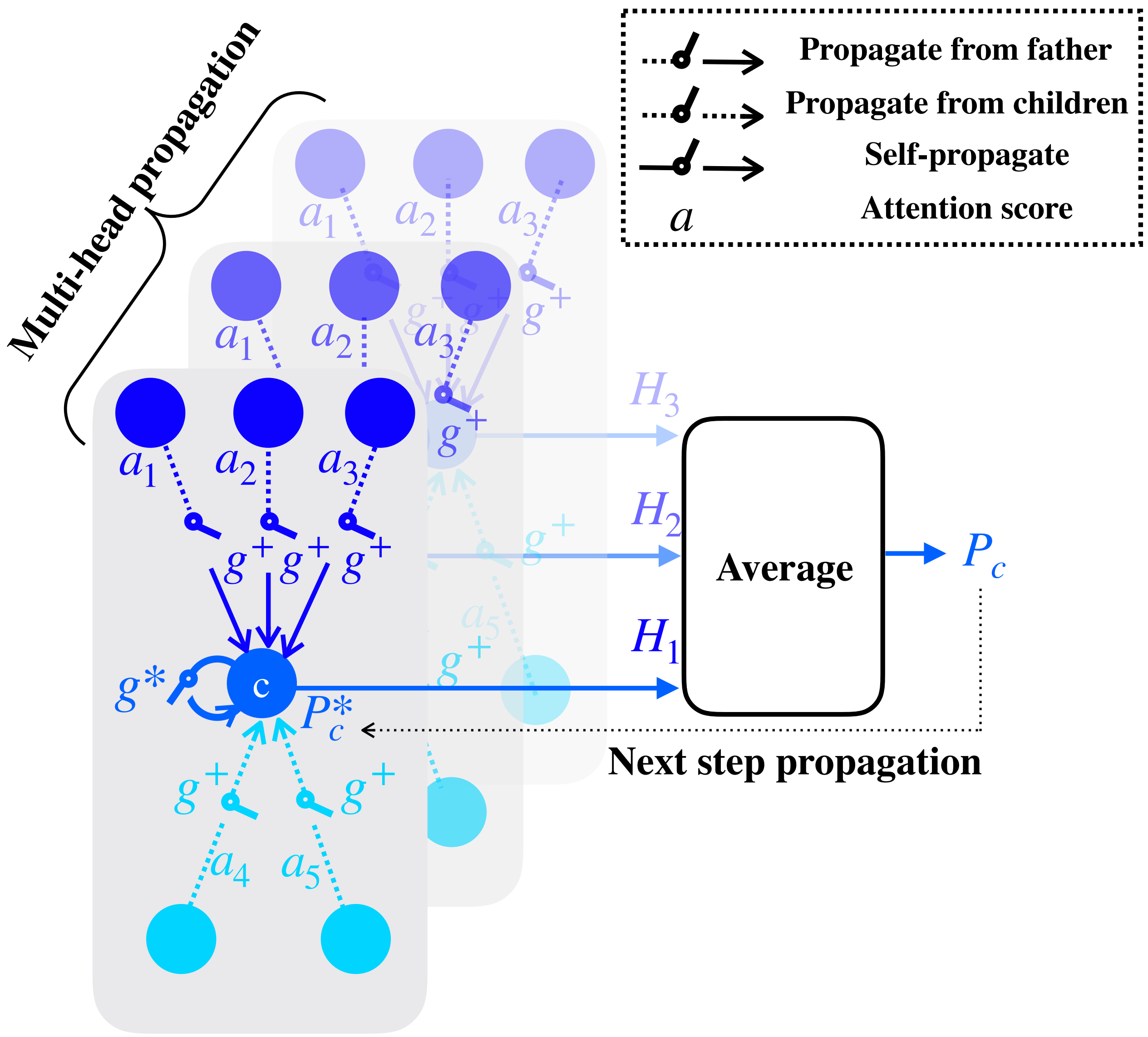
2. **Aggregated messages from its neighbors**

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.

$$\boldsymbol{P}_y^{t+1} \triangleq g\boldsymbol{P}_{y\to y}^{t+1} + (1-g)\boldsymbol{P}_{\mathcal{N}_y\to y}^{t+1},$$

$$g = \frac{\exp[\gamma\cos(\boldsymbol{P}_y^0, \boldsymbol{P}_{y\to y}^{t+1})]}{\exp[\gamma\cos(\boldsymbol{P}_y^0, \boldsymbol{P}_{y\to y}^{t+1})] + \exp[\gamma\cos(\boldsymbol{P}_y^0, \boldsymbol{P}_{\mathcal{N}_y\to y}^{t+1})]},$$

Gated Propagation Networks (Cont.)



1. **Initialize prototype**

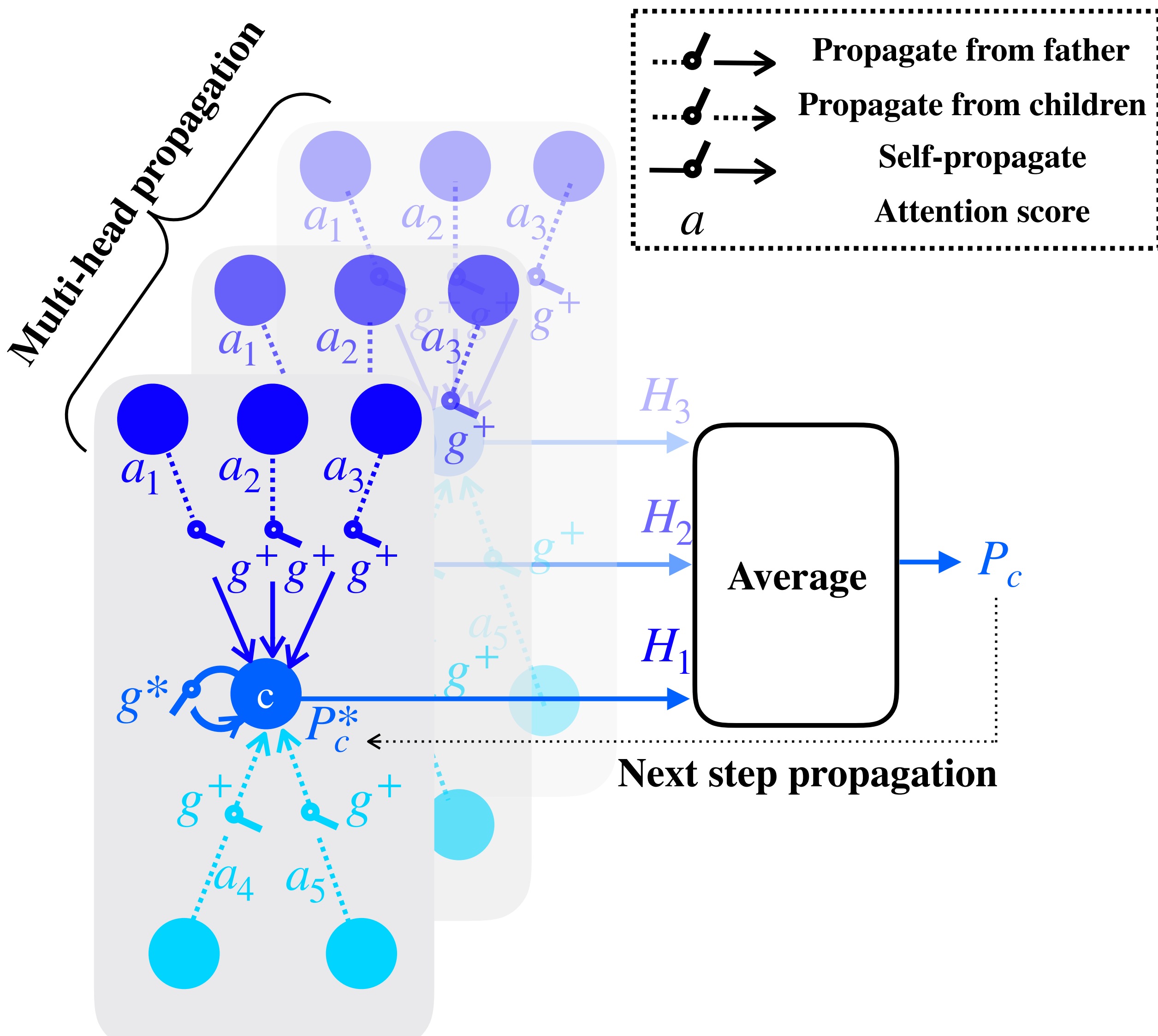
2. **Aggregated messages from its neighbors**

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.

$$\boldsymbol{P}_y^{t+1} = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{P}_y^{t+1}[i]$$

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:

$$\boldsymbol{P}_y \triangleq \lambda \times \boldsymbol{P}_y^0 + (1-\lambda) \times \boldsymbol{P}_y^T$$

Training Strategy

- **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

Training Strategy (Cont.)

Algorithm 1 GPN Training

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

Initialization: Θ^{cnn}, Θ^{prop}, Θ^{fc}, $\tau \leftarrow 0$;

for $\tau \in \{1, \cdots, \tau_{total}\}$ **do**

  if $\tau \bmod m = 0$ **then**

    Update prototypes in memory by Eq. (3);

  end if

  Draw $\alpha \sim$Unif$[0, 1]$;

  if $\alpha < 0.9^{20\tau/\tau_t}$ **then**

    Train a classifier to update Θ^{cnn} with loss $\sum_{(\boldsymbol{x},y)\sim\mathcal{D}} - \log \Pr(y|\boldsymbol{x}; \Theta^{cnn}, \Theta^{fc})$;

  else

    Sample a few-shot task T as in Sec. 3.3;

    Construct MST $\mathcal{Y}^T_{MST}$ as in Sec. 3.3;

    For $y \in \mathcal{Y}^T_{MST}$, compute $\boldsymbol{P}^0_y$ by Eq. (3) if $y \in T$, otherwise fetch $\boldsymbol{P}^0_y$ from memory;

    for $t \in \{1, \cdots, \mathcal{T}\}$ **do**

      For all $y \in \mathcal{Y}^T_{MST}$, concurrently update their prototypes $\boldsymbol{P}^t_y$ by Eq. (4)-(6);

    end for

    Compute $\boldsymbol{P}_y$ for $y \in \mathcal{Y}^T_{MST}$ by Eq.(7);

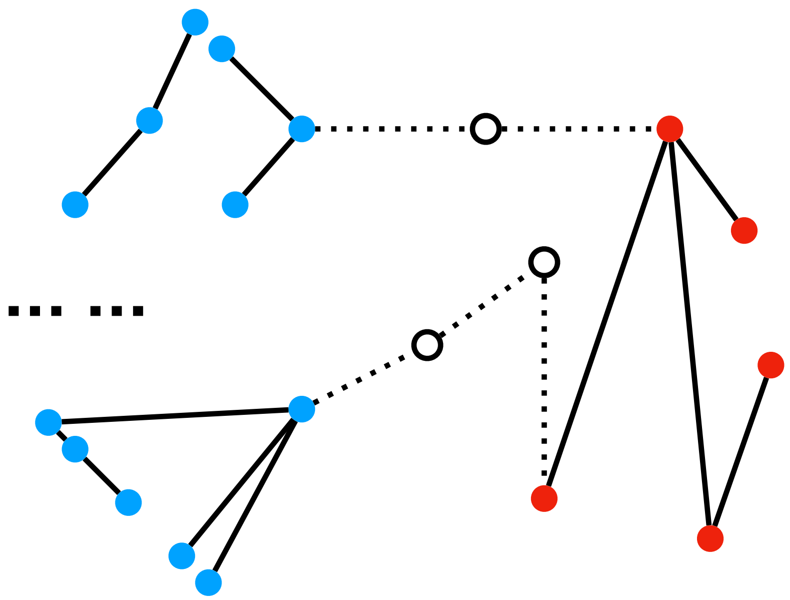
    Compute $\log \Pr(y|\boldsymbol{x}; \Theta^{cnn}, \Theta^{prop})$ by Eq. (2) for all samples $(\boldsymbol{x}, y)$ in task T;

    Update Θ^{cnn} and Θ^{prop} by minimizing $\sum_{(x,y)\sim\mathcal{D}^T} - \log \Pr(y|x; \Theta^{cnn}, \Theta^{prop})$;

  end if

end for

Experimental Results



tieredImageNet-Close

1~4 steps/hops between training classes and test classes

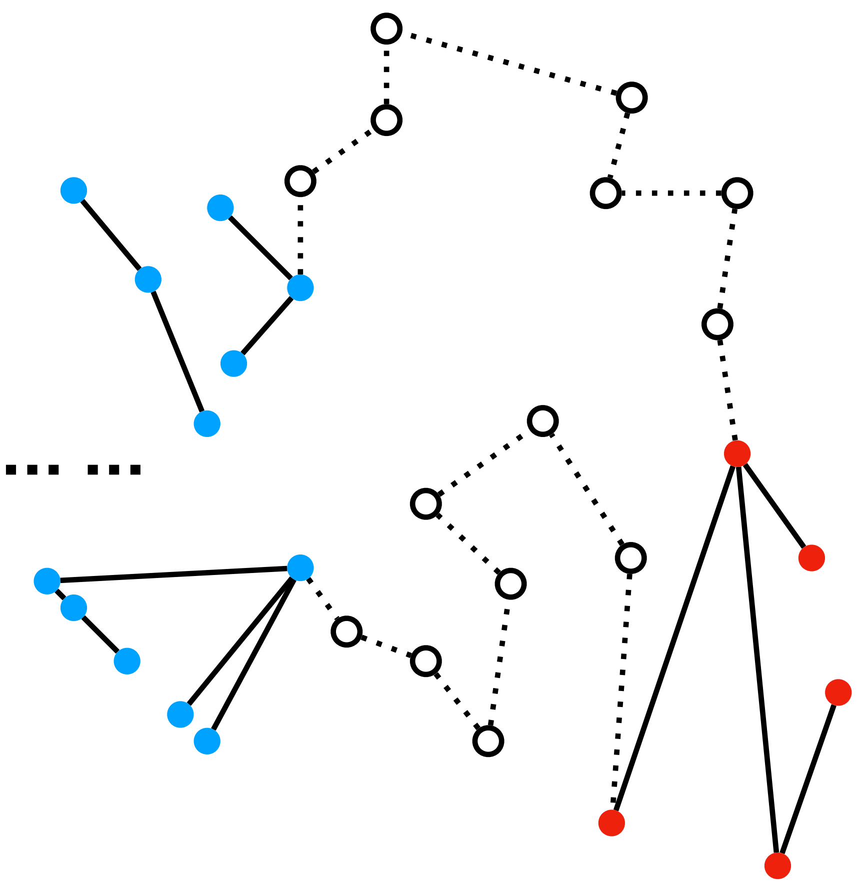
Table 3: Validation accuracy (mean±CI%95) on 600 test tasks achieved by GPN and baselines on *tiered*ImageNet-**Close** with few-shot tasks generated by **random sampling**.

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

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

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

Experimental Results



tieredImageNet-Far

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

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

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

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

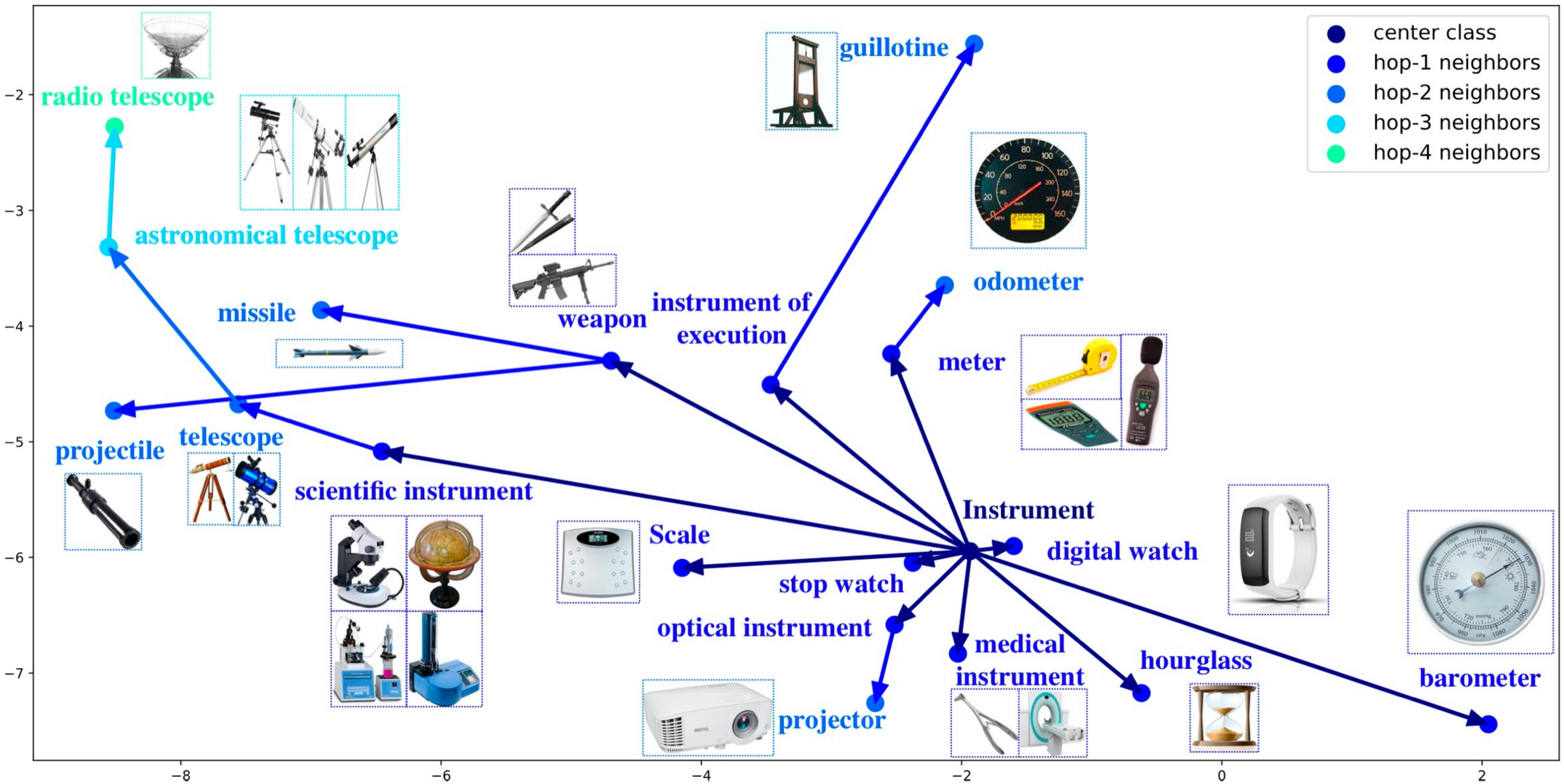
Model	**5way1shot**	**5way5shot**	**10way1shot**	**10way5shot**
Prototypical Net [23]	43.57±1.67%	62.35±1.06%	29.88±1.11%	46.48±0.70%
GNN [6]	44.00±1.36%	62.00±0.66%	28.50±0.60%	46.17±0.74%
Closer Look [3]	38.37±1.57%	54.64±0.85%	30.40±1.09%	33.72±0.43%
PPN [15]	42.40±1.63%	61.37±1.05%	28.67±1.01%	46.02±0.61%
GPN	**47.74±1.76%**	**63.53±1.03%**	**32.94±1.16%**	**47.43±0.67%**
GPN+	**47.58±1.70%**	**63.74±0.95%**	**32.68±1.17%**	**47.44±0.71%**

Ablation Studies

Table 7: Validation accuracy (mean±CI%95) for possible variants of GPN on *tiered*ImageNet-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.

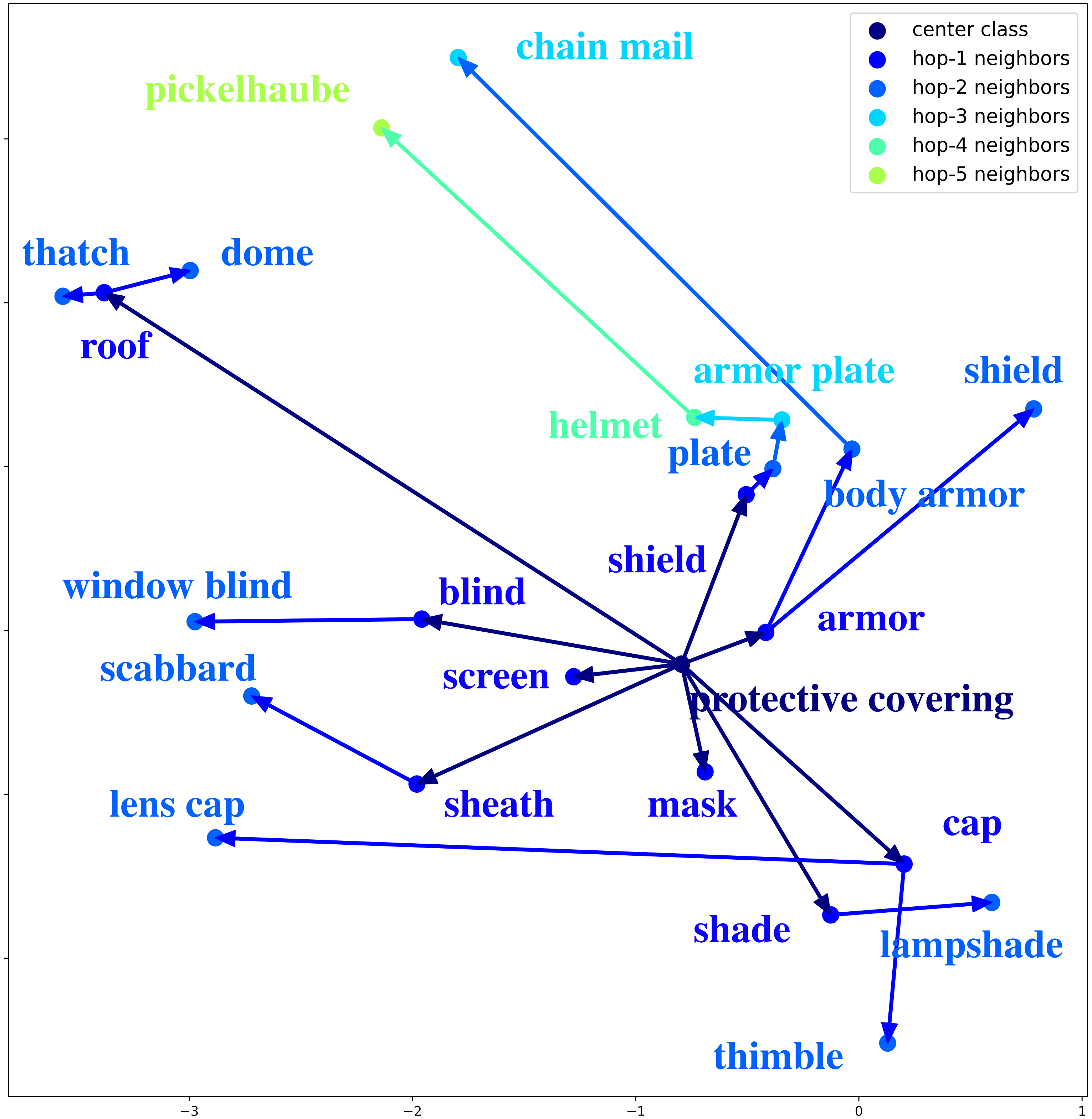
Task Generation			**Propagation Mechanism**					**Training**		**Model**			**ACCURACY**
SR-S	S-S	R-S	N$\rightarrow$**C**	F$\rightarrow$C	C$\rightarrow$C	B$\rightarrow$P	M$\rightarrow$P	**AUX**	**MST**	**M-H**	**M-A**	A-A	
	✓		✓					✓	✓	✓	✓		46.20±1.70%
		✓	✓					✓	✓	✓	✓		49.33±1.68%
✓				✓				✓	✓	✓	✓		42.60±1.61%
✓					✓			✓	✓	✓	✓		37.90±1.50%
✓						✓		✓	✓	✓	✓		47.90±1.72%
✓							✓	✓	✓	✓	✓		46.90±1.78%
✓			✓						✓	✓	✓		41.87±1.72%
✓			✓					✓		✓	✓		45.83±1.64%
✓			✓					✓	✓		✓		49.40±1.69%
✓			✓					✓	✓	✓		✓	46.74±1.71%
✓			✓					✓	✓	✓	✓		**50.54**±**1.67**%

Visualization



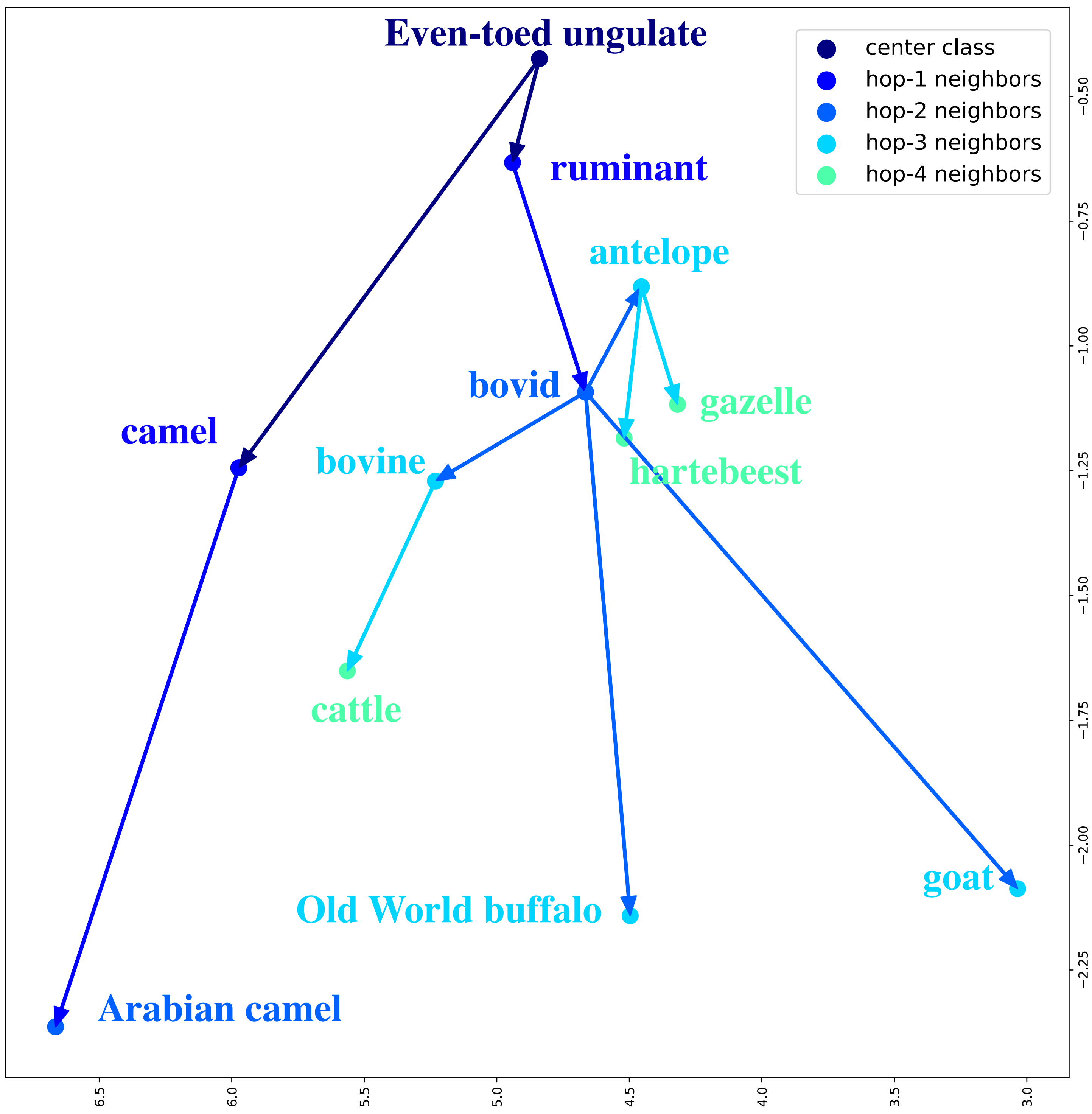
Visualization of the class prototypes produced by GPN for few-shot tasks and the associated graph.

Visualization (Cont.)



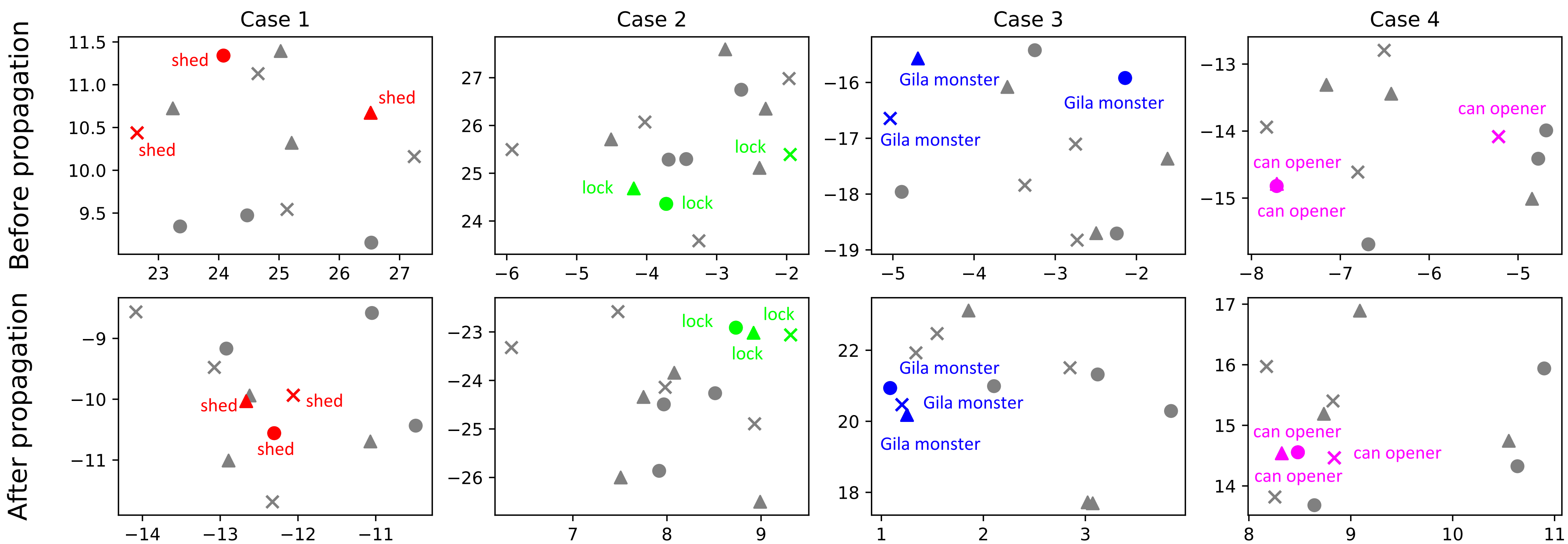
More …

Visualization (Cont.)



More …

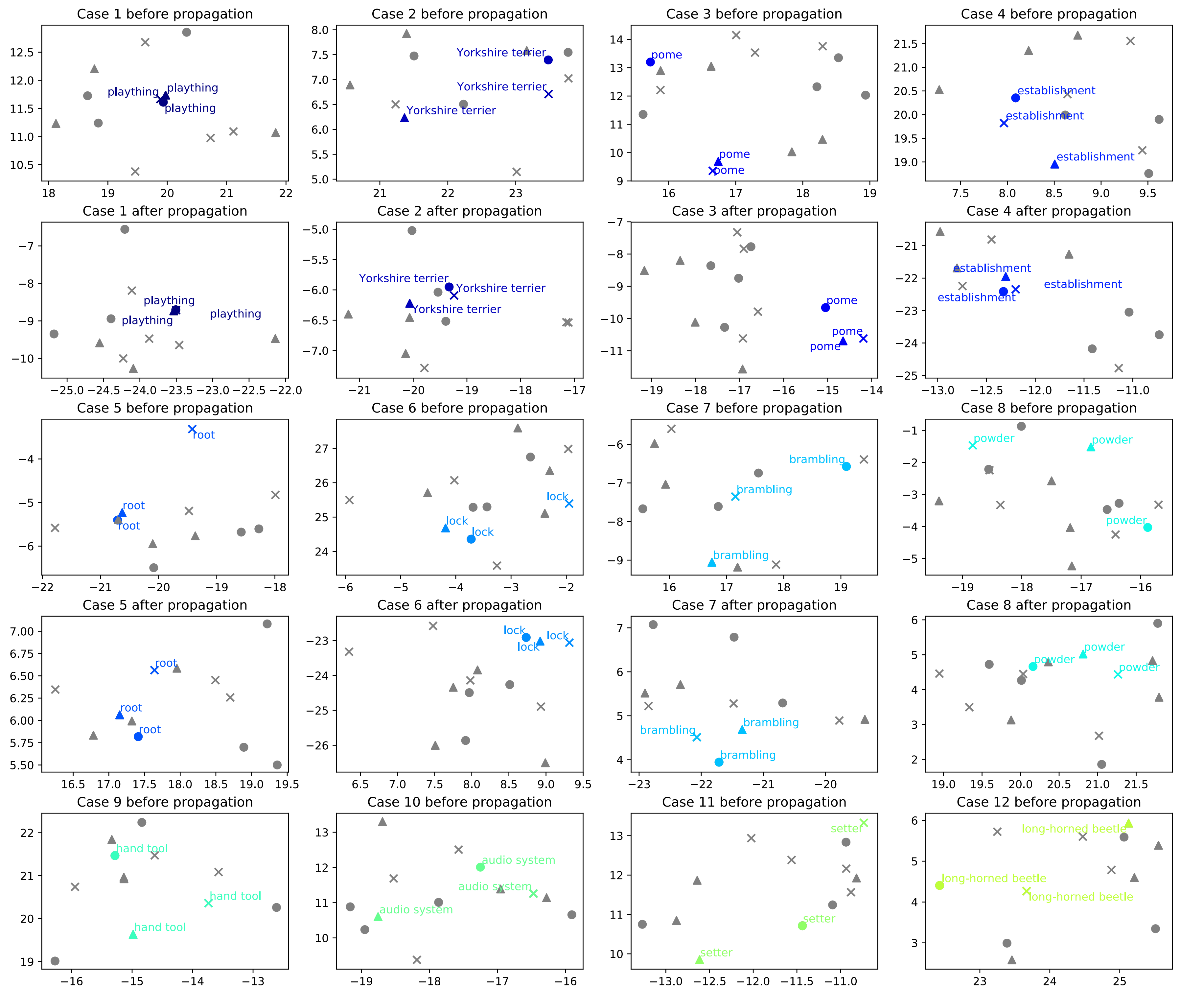
Visualization (Cont.)



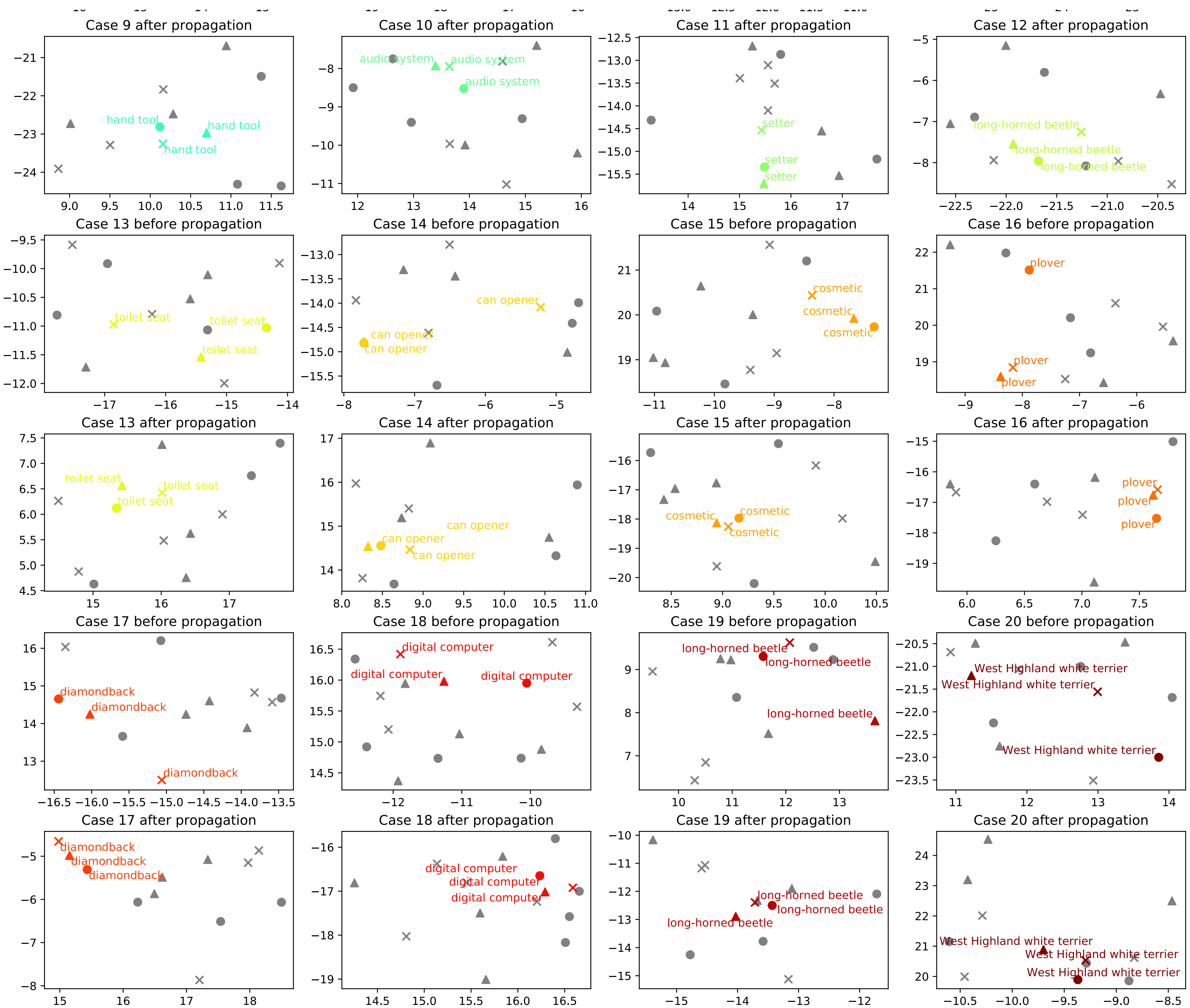
Prototypes **before** (top row) and **after GPN propagation** (bottom row) on tieredImageNet-Close by random sampling for 5-way-1-shot few-shot learning. The prototypes in top row equal to the ones achieved by prototypical network. Different tasks are marked by a different shape, and classes shared by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



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.

For more, refer our paper at https://arxiv.org/abs/1909.05024