KDD-23 Research Track Paper

Task-Equivariant Graph Few-shot Learning

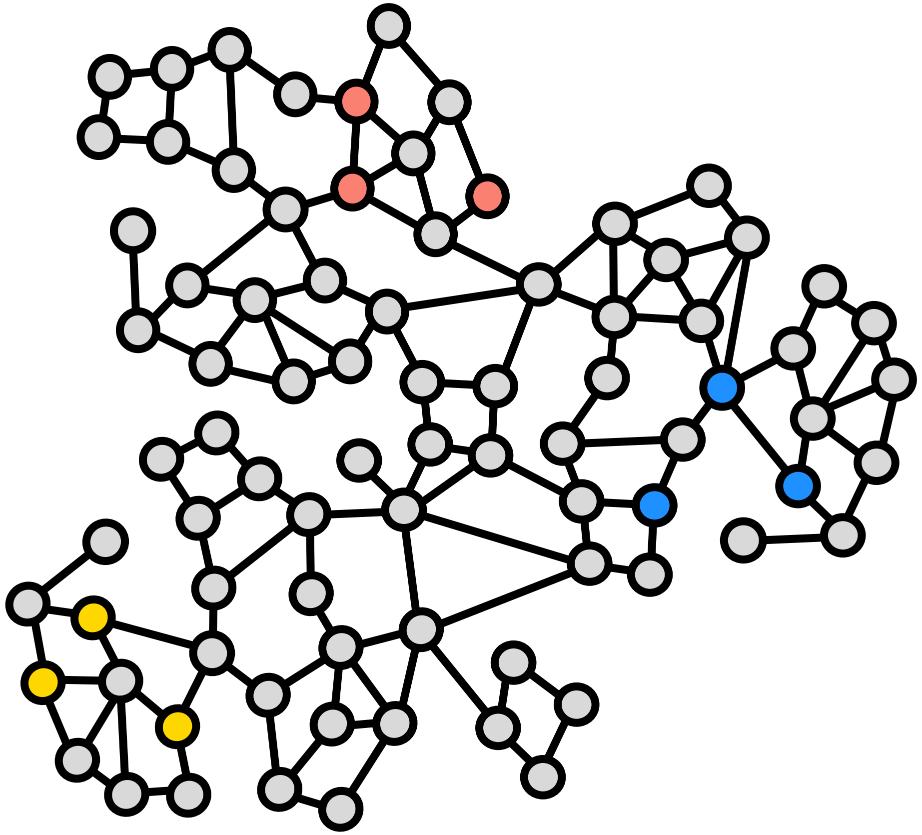
Sungwon Kim, Junseok Lee, Namkyeong Lee,
Wonjoon Kim, Seungyoon Choi, Chanyoung Park

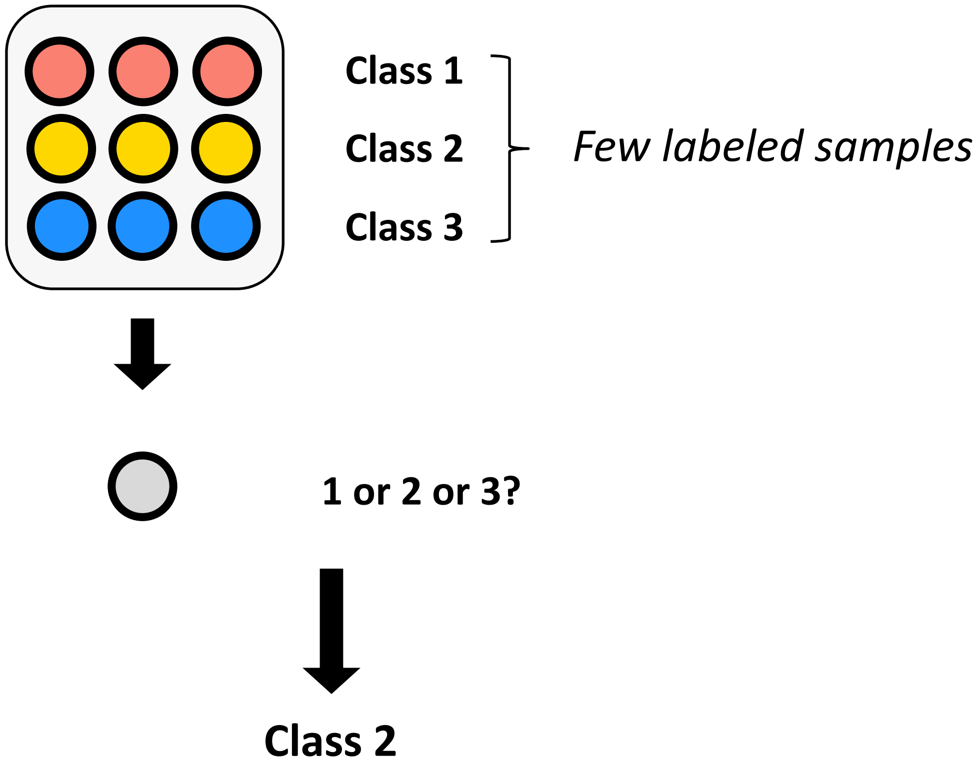
Korea Advanced Institute of Science and Technology (KAIST)

AUGUST 6 - 10
KDD2023
LONG BEACH, CA

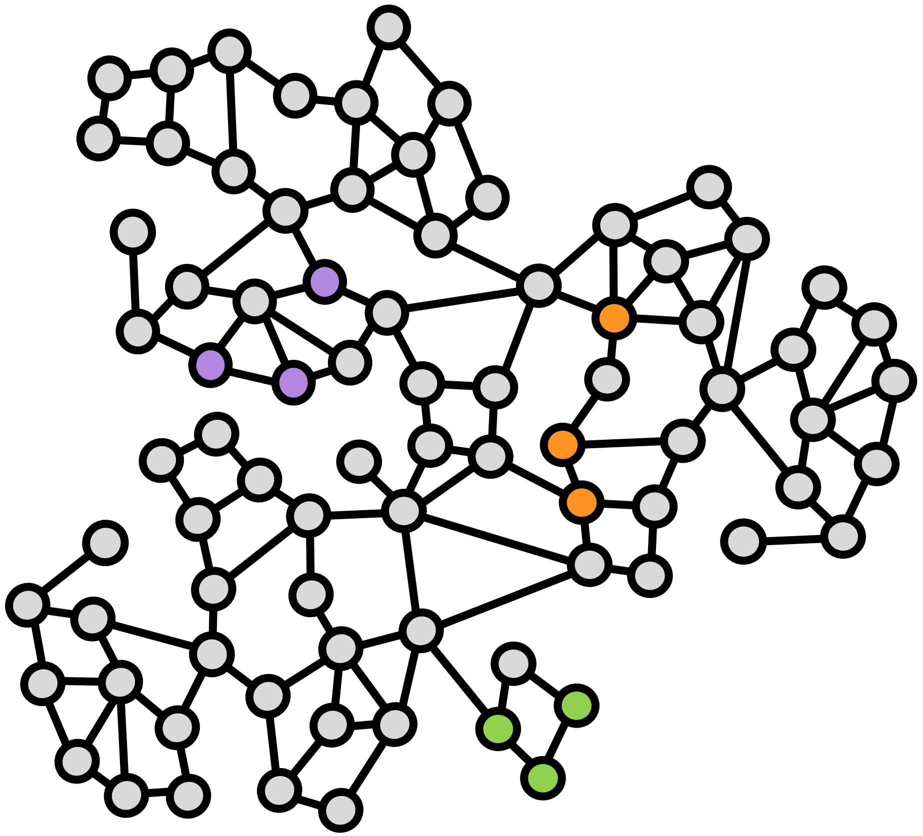
DSAIL @ KAIST

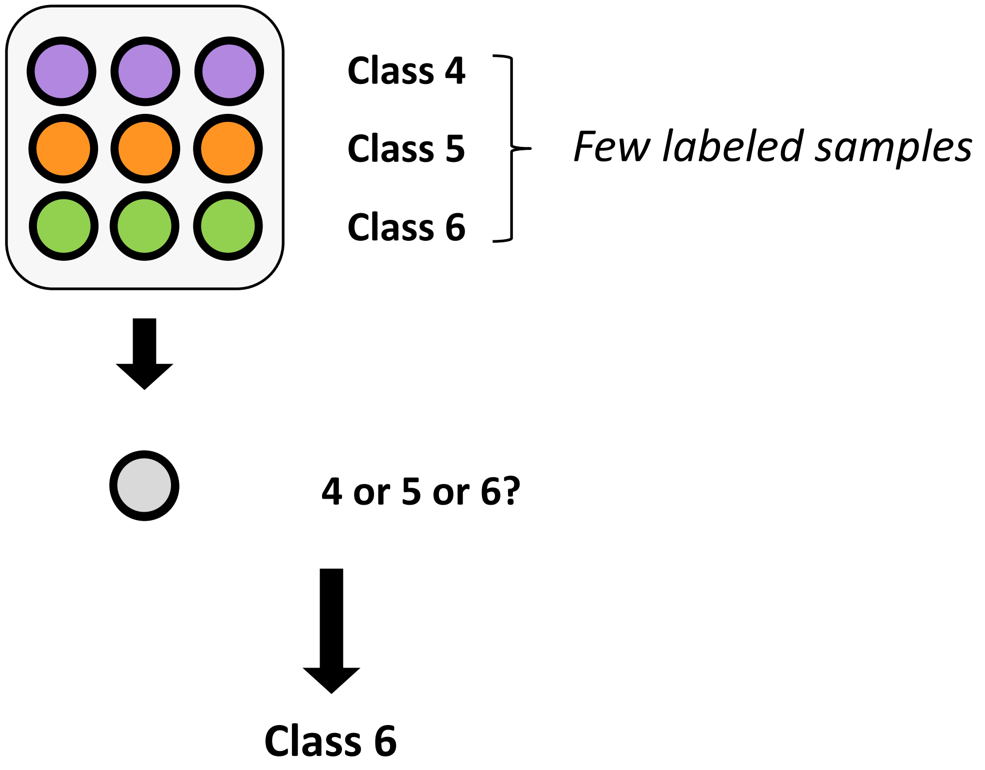
FEW-SHOT LEARNING



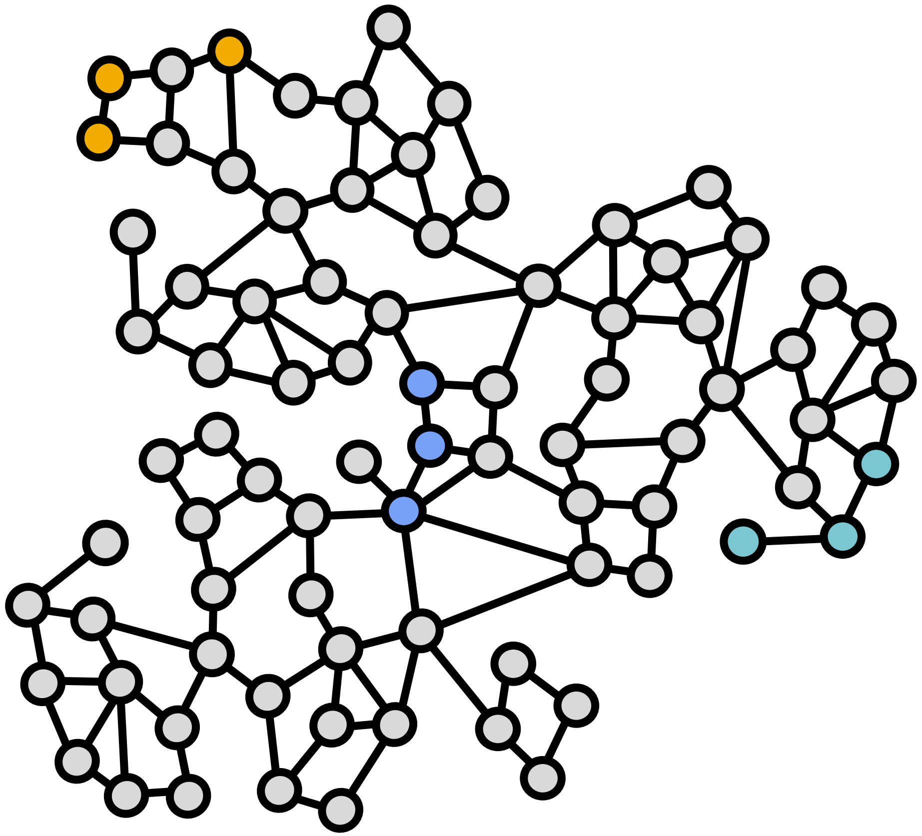


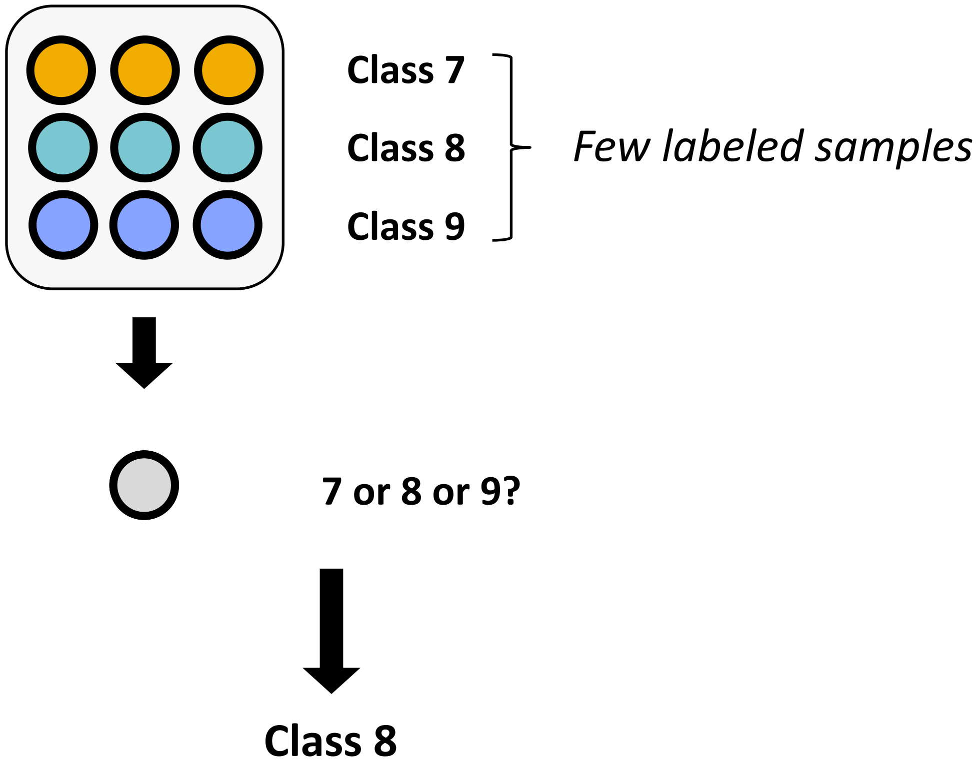
FEW-SHOT LEARNING



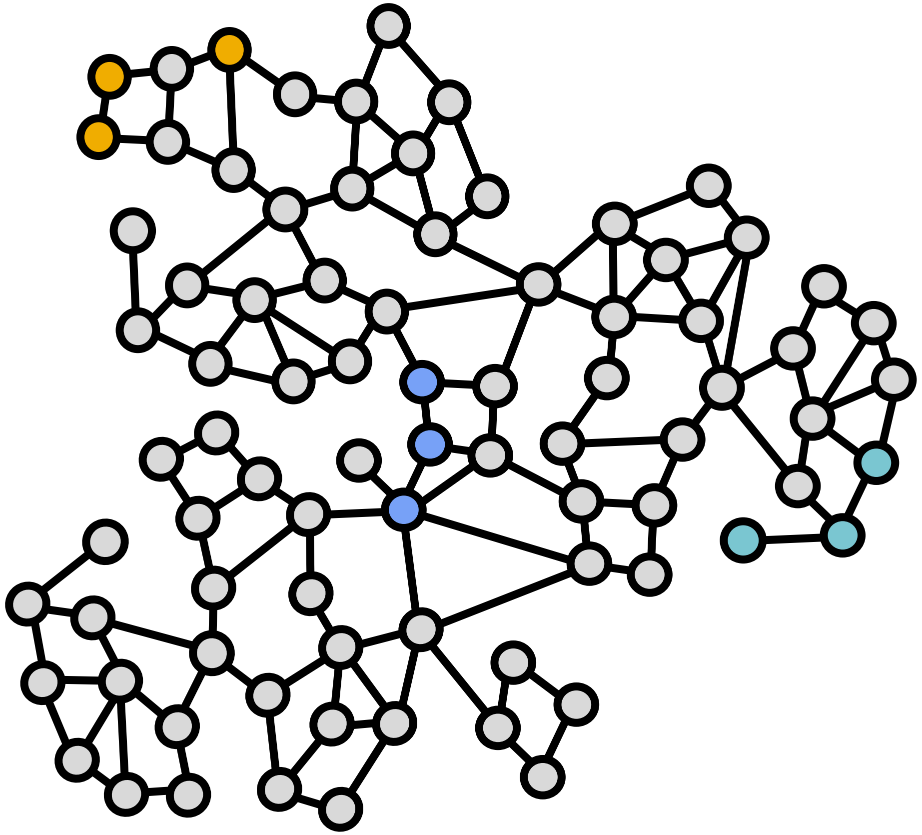


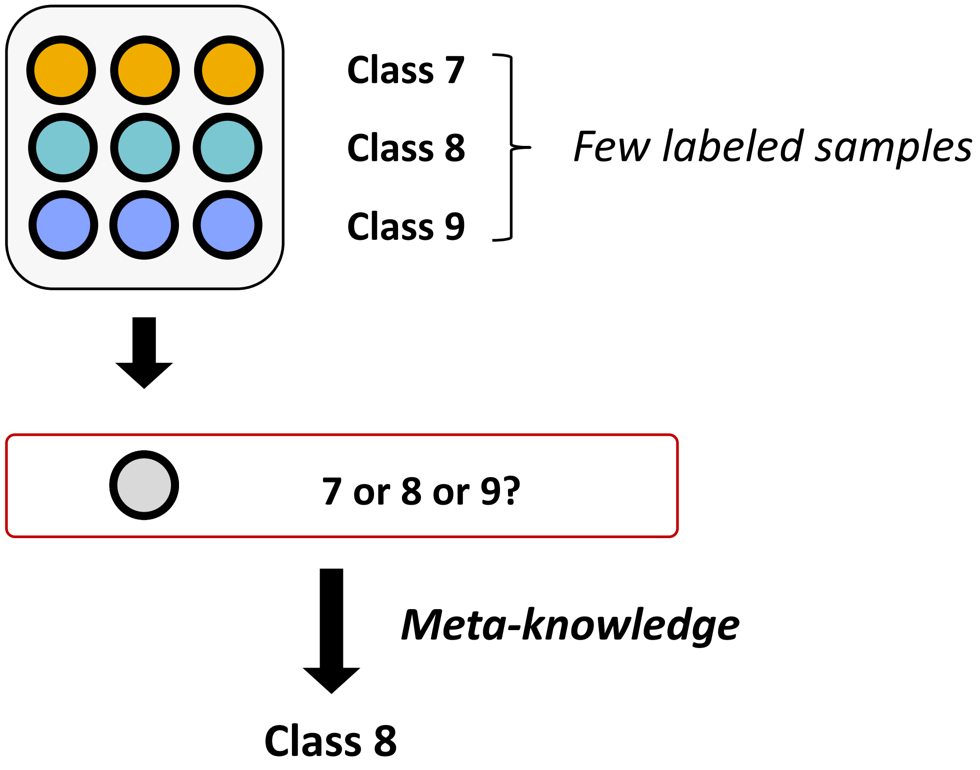
FEW-SHOT LEARNING



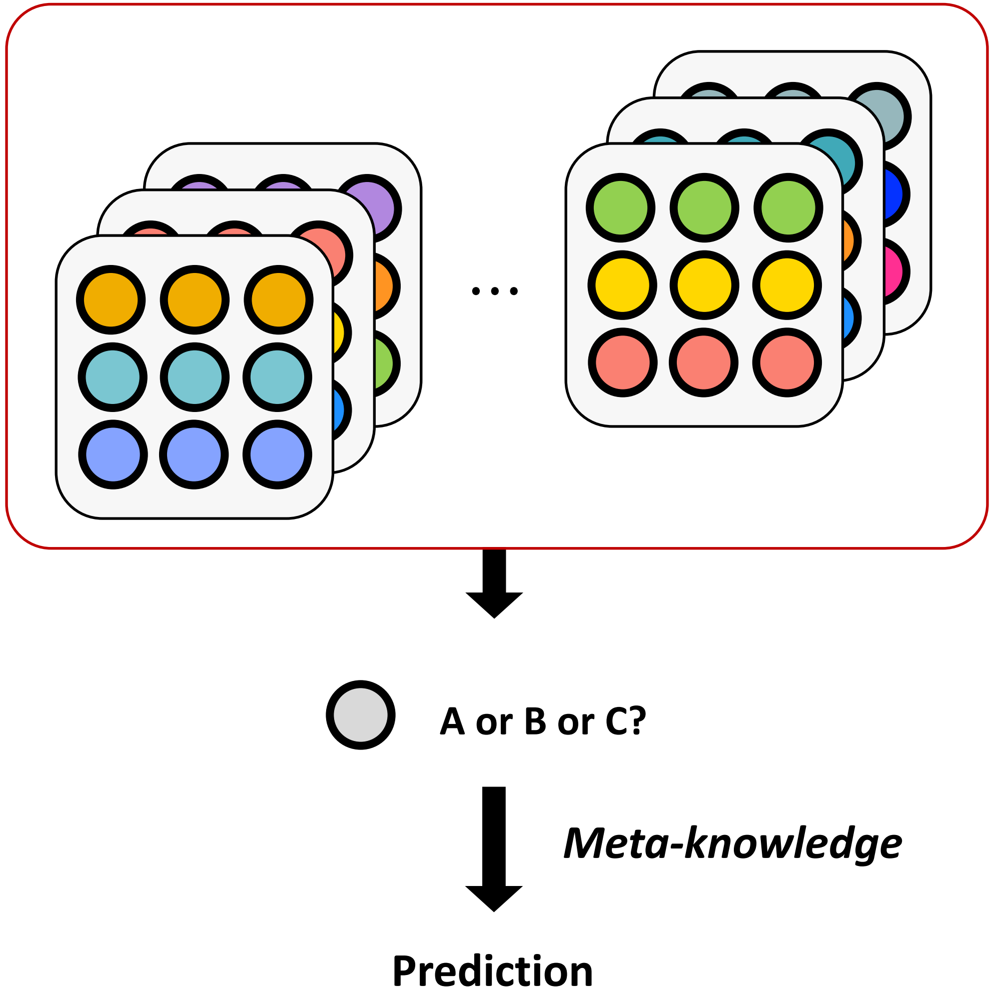


FEW-SHOT LEARNING





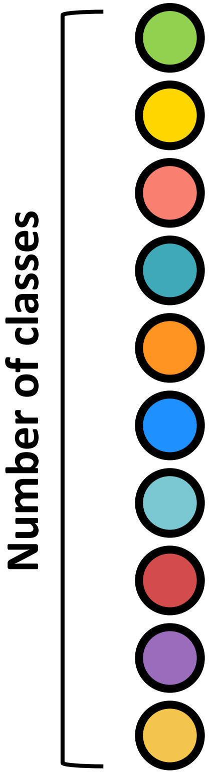
FEW-SHOT LEARNING



To ensure the strong generalization power of meta-knowledge,
a significant number of training tasks are needed!

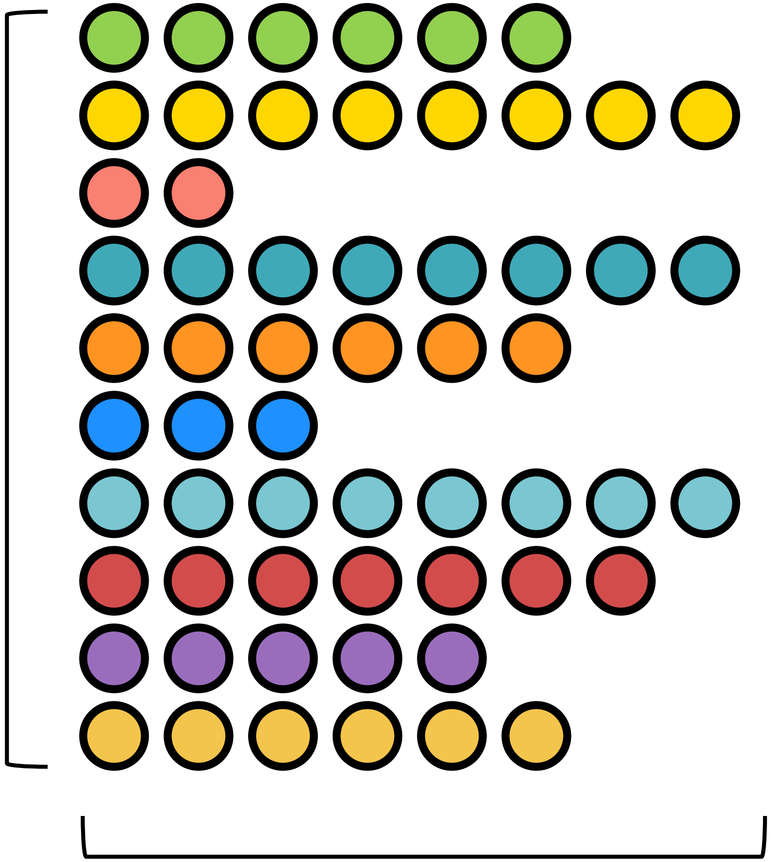
IMPACT OF DIVERSITY OF TRAIN TASKS

Training data



IMPACT OF DIVERSITY OF TRAIN TASKS

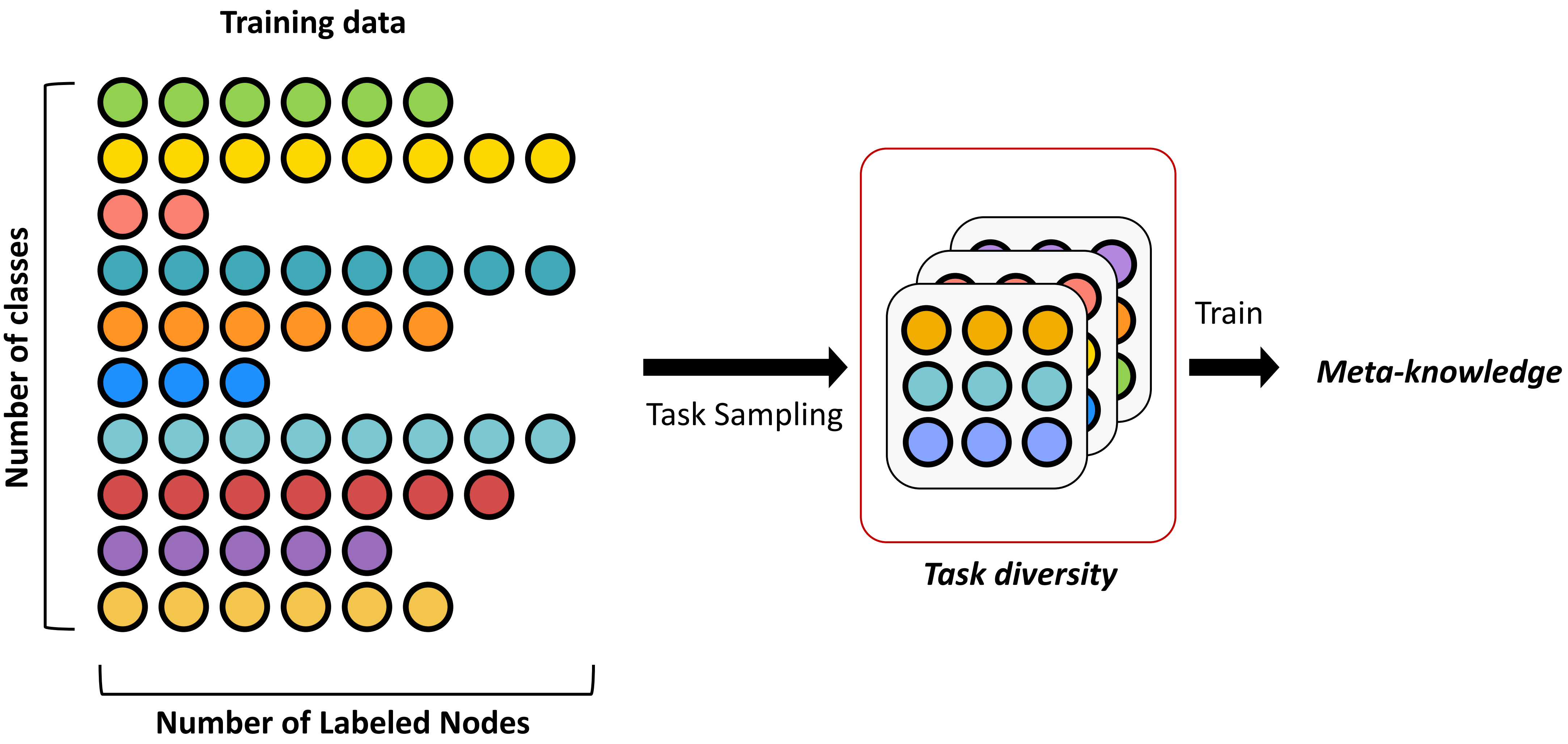
Training data



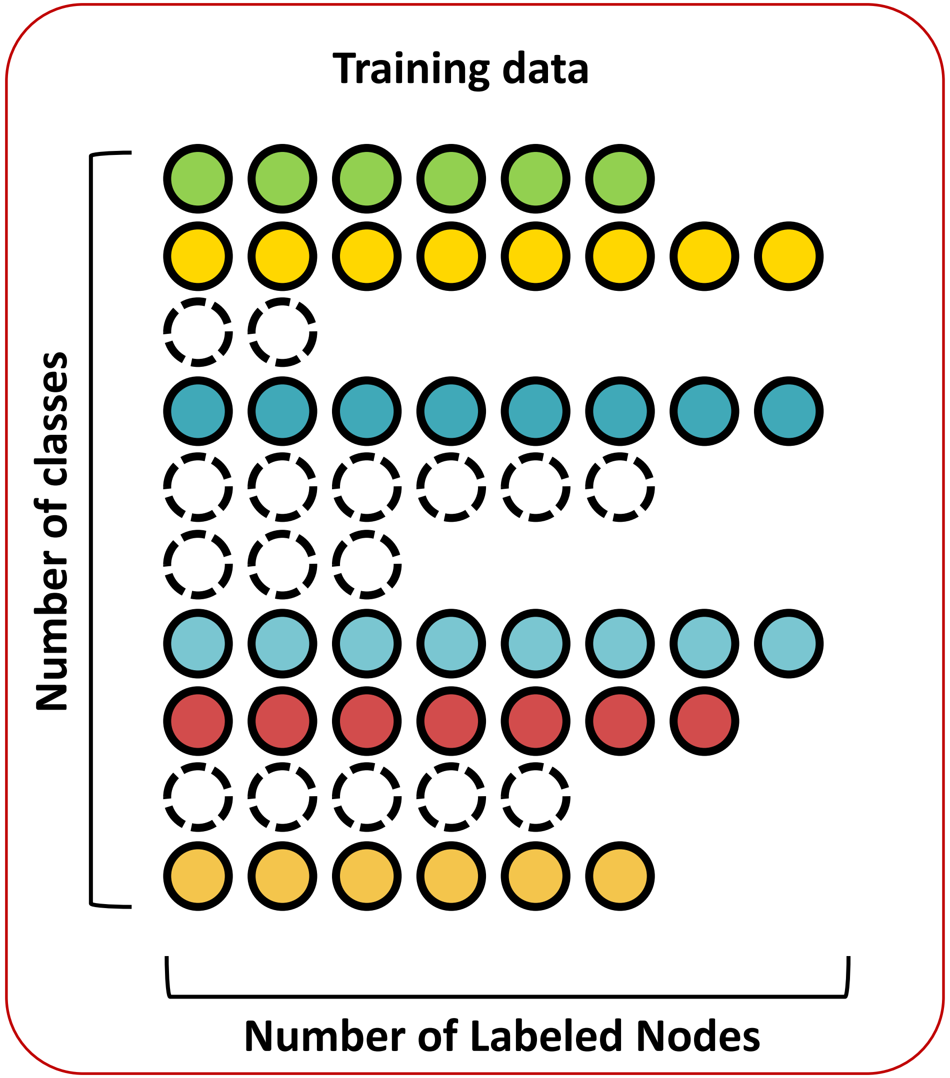
Number of classes

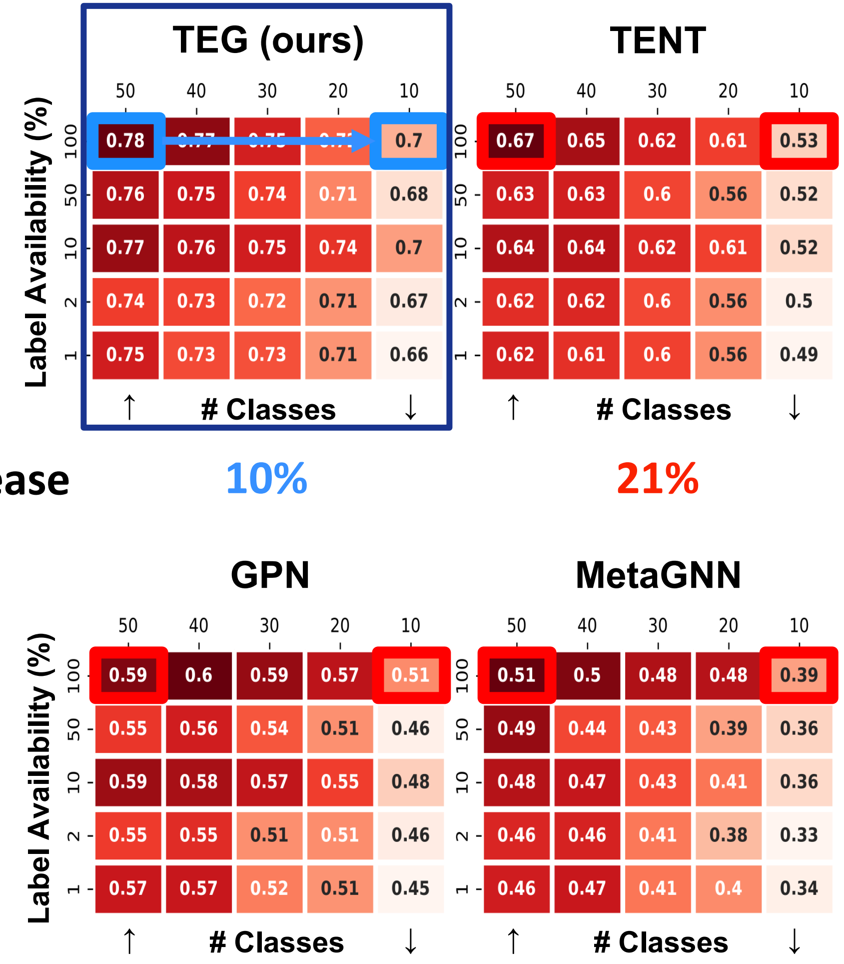
Number of Labeled Nodes

IMPACT OF DIVERSITY OF TRAIN TASKS



IMPACT OF DIVERSITY OF TRAIN TASKS



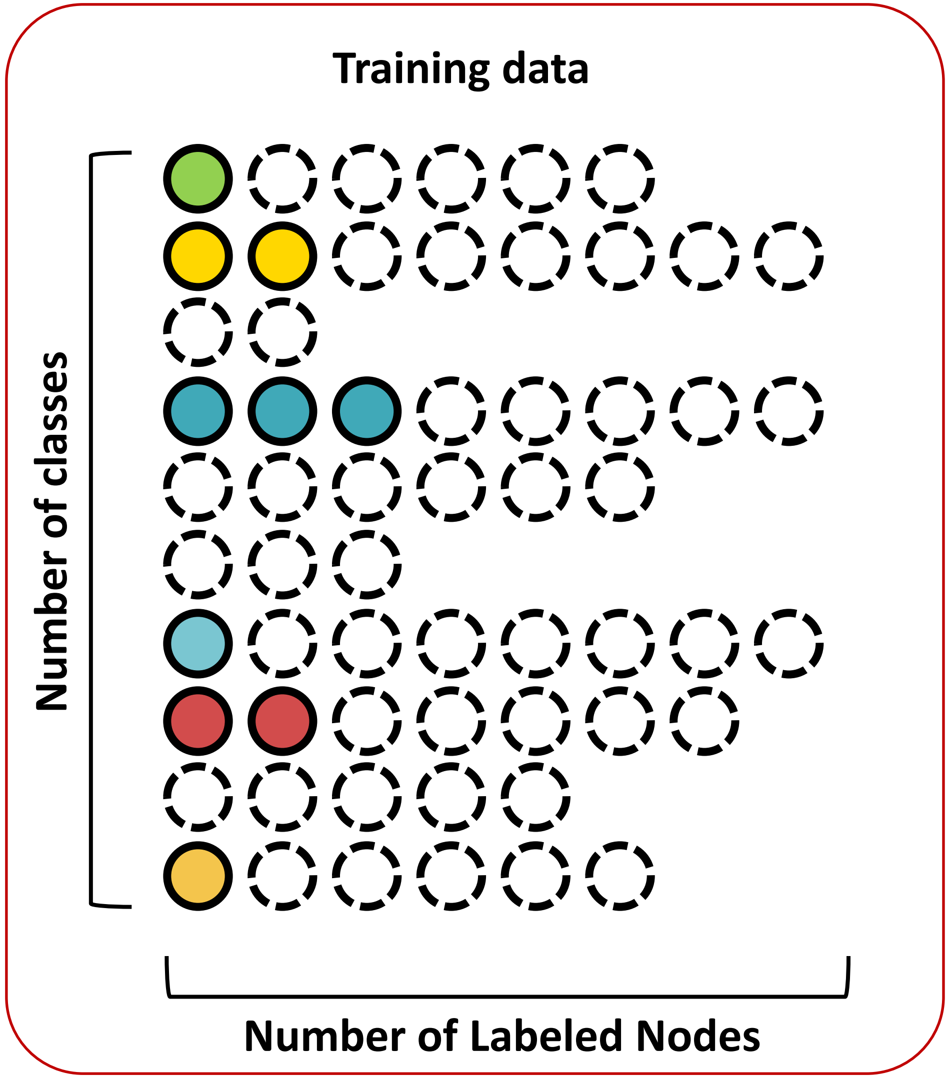


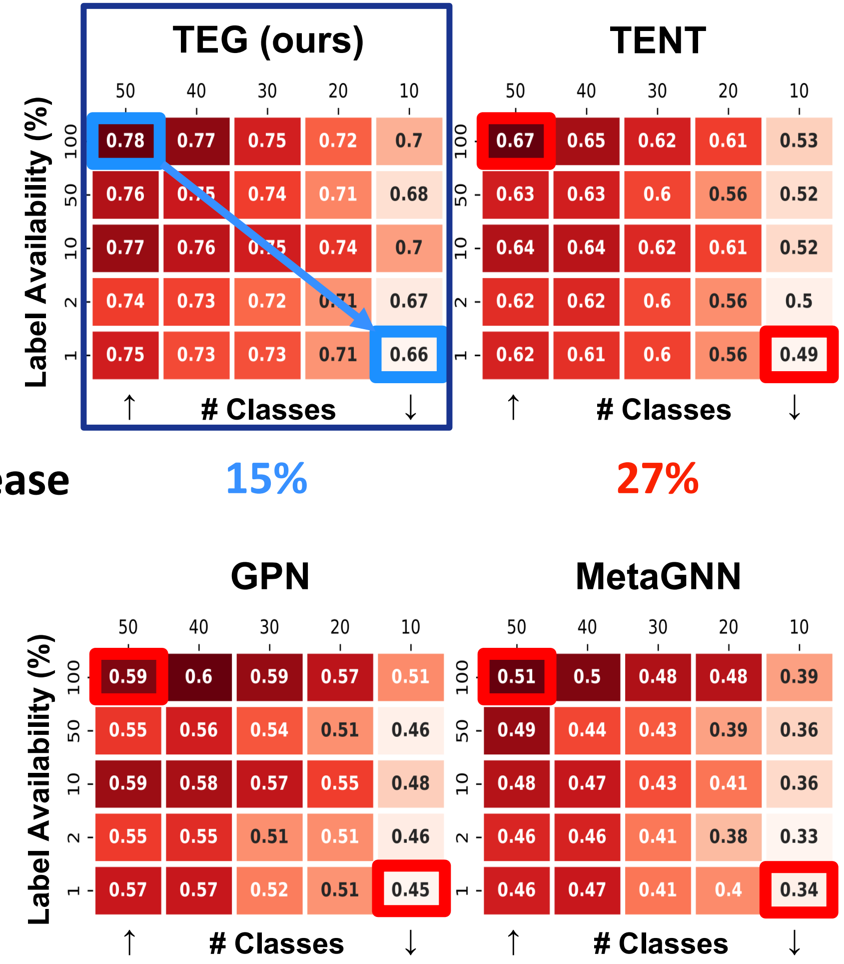
% Decrease 10% 21%

% Decrease 14% 24%

Performance (%) in 10way 5shot meta-training tasks. Amazon-electronics dataset is used.

IMPACT OF DIVERSITY OF TRAIN TASKS



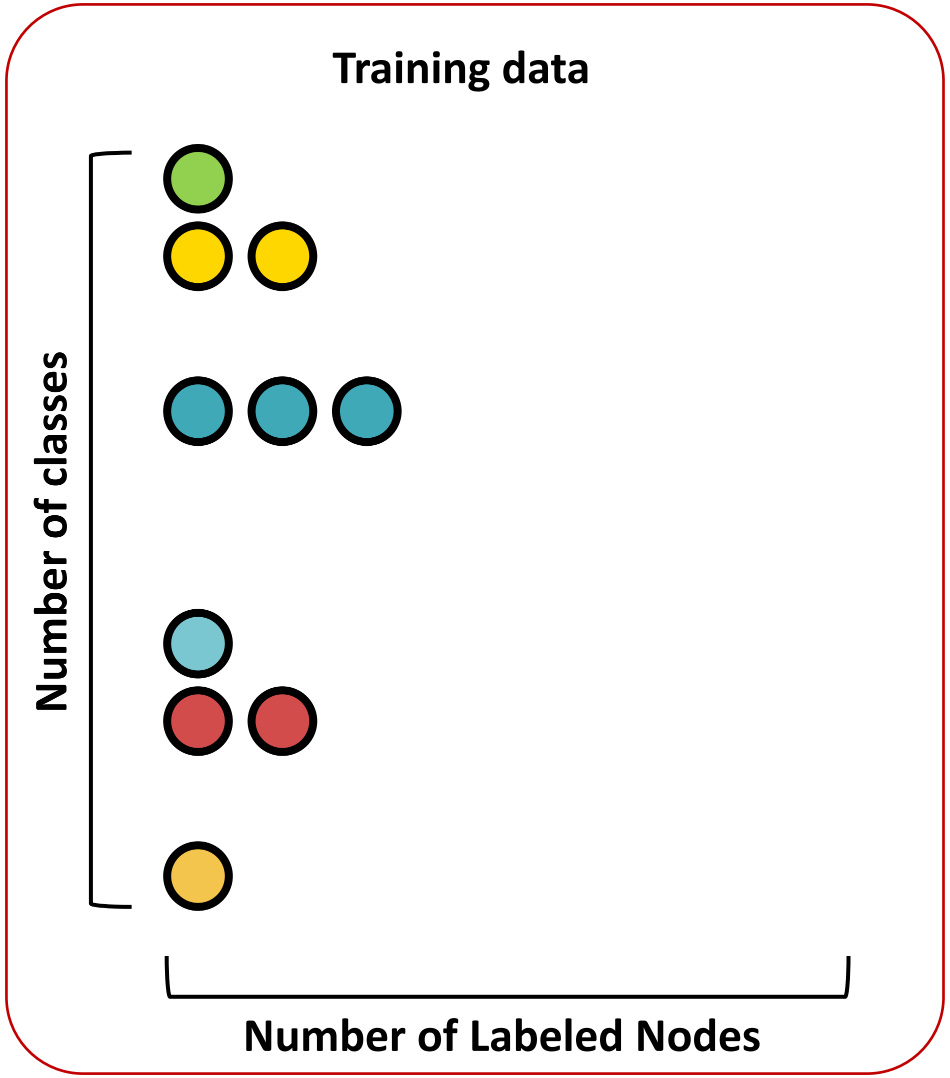


% Decrease 15% 27%

% Decrease 24% 33%

Performance (%) in 10way 5shot meta-training tasks. Amazon-electronics dataset is used.

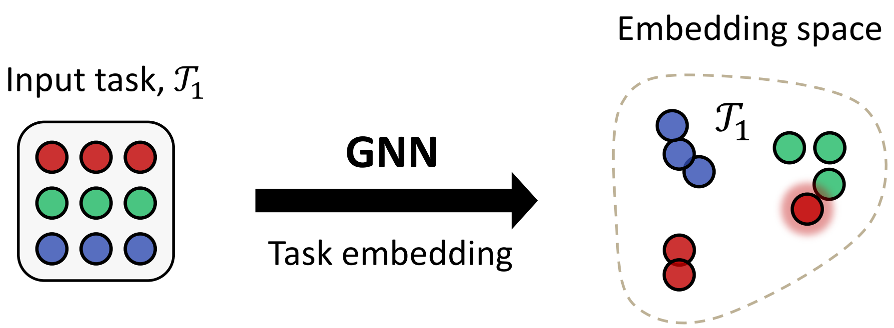
IMPACT OF DIVERSITY OF TRAIN TASKS



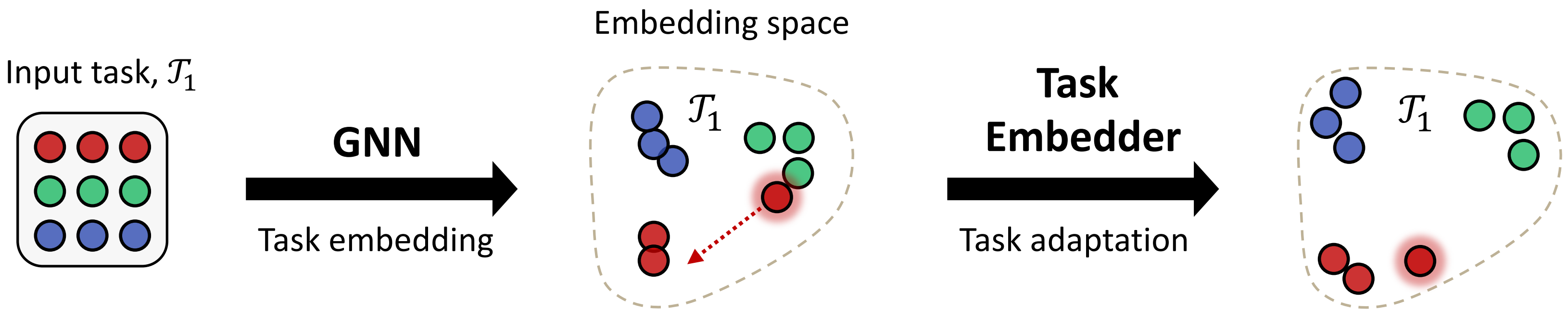
In real-world scenarios, ***creating diverse tasks becomes challenging*** due to the ***high cost of labeling.***

TEG learns ***highly transferable*** *meta-knowledge* ***with limited diversity*** *of training tasks!*

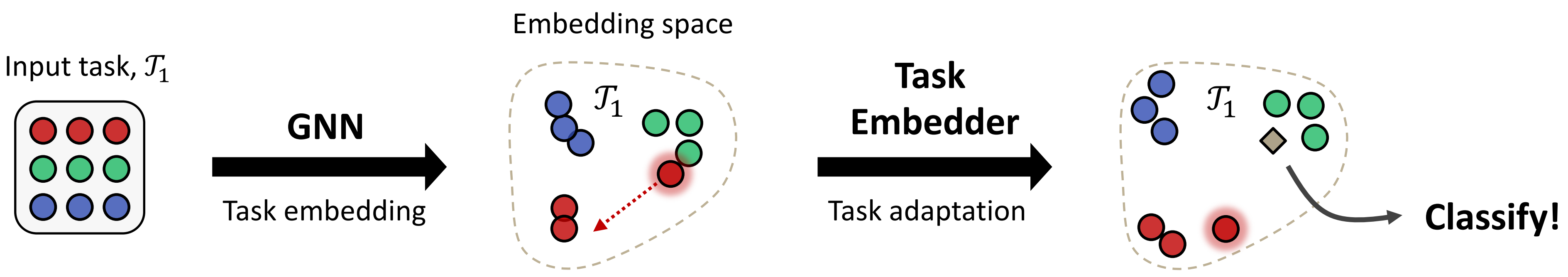
THE GENERAL PROCESS FOR SOLVING THE TASK



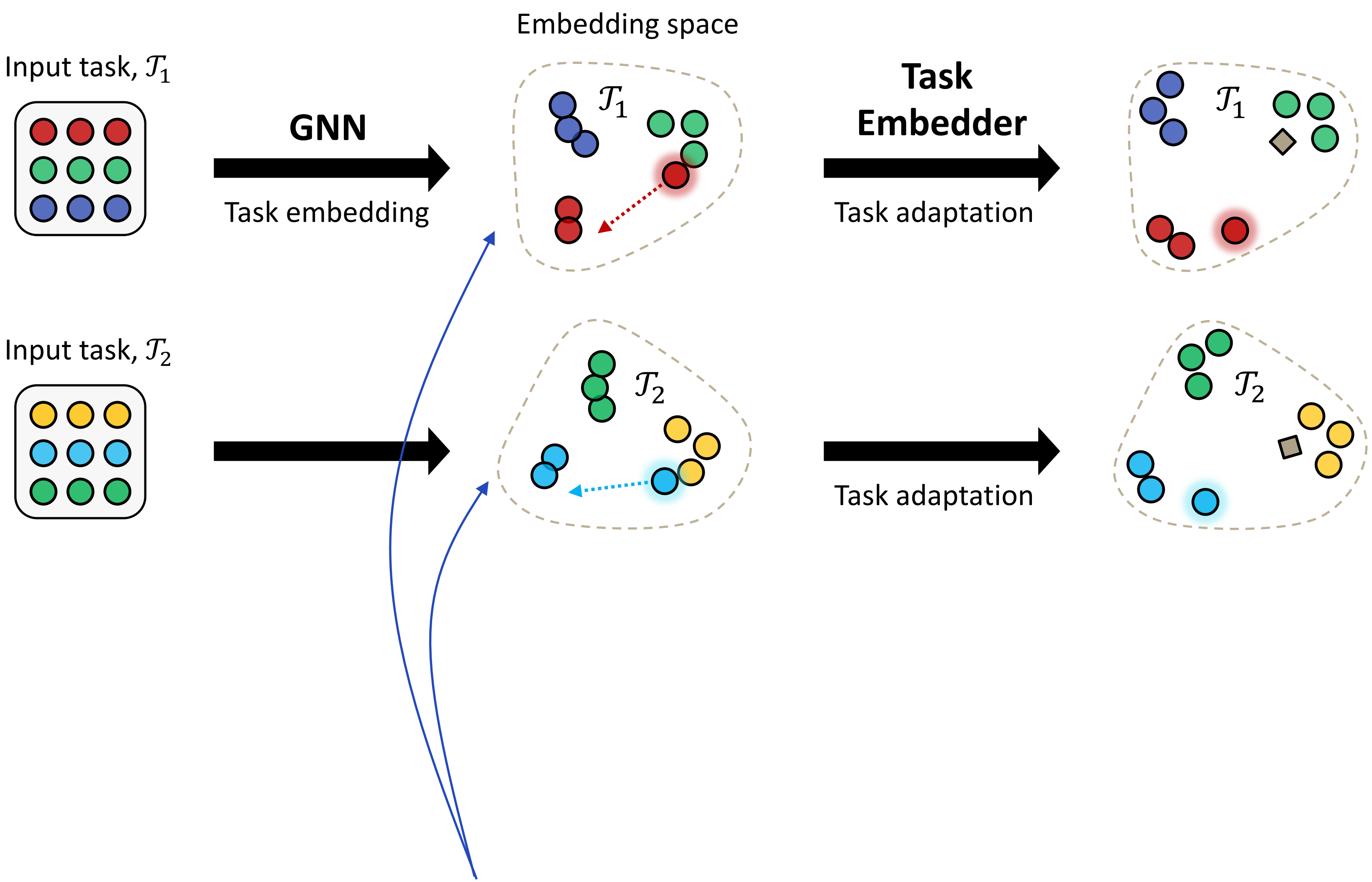
THE GENERAL PROCESS FOR SOLVING THE TASK



THE GENERAL PROCESS FOR SOLVING THE TASK

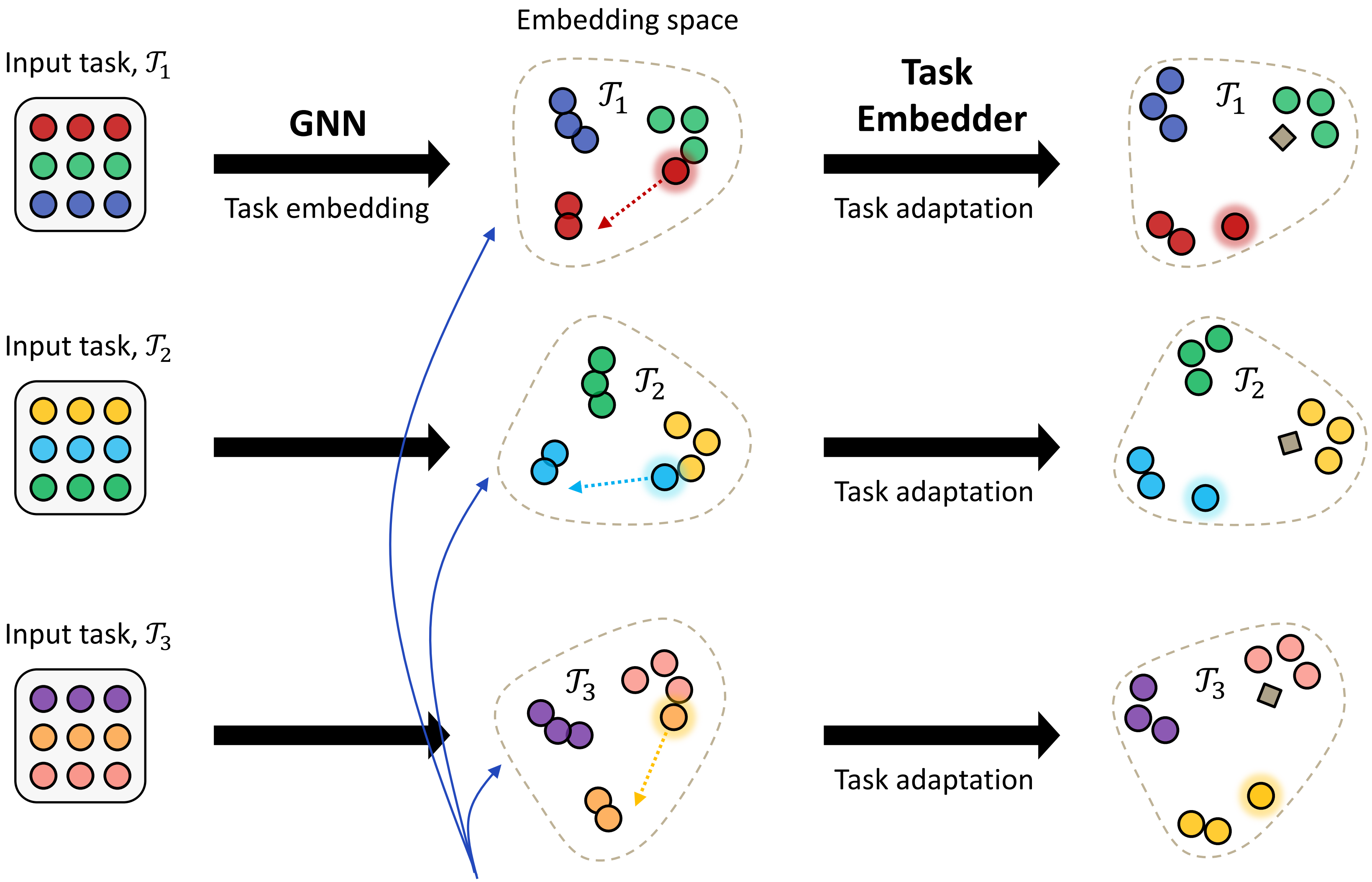


THE GENERAL PROCESS FOR SOLVING THE TASK



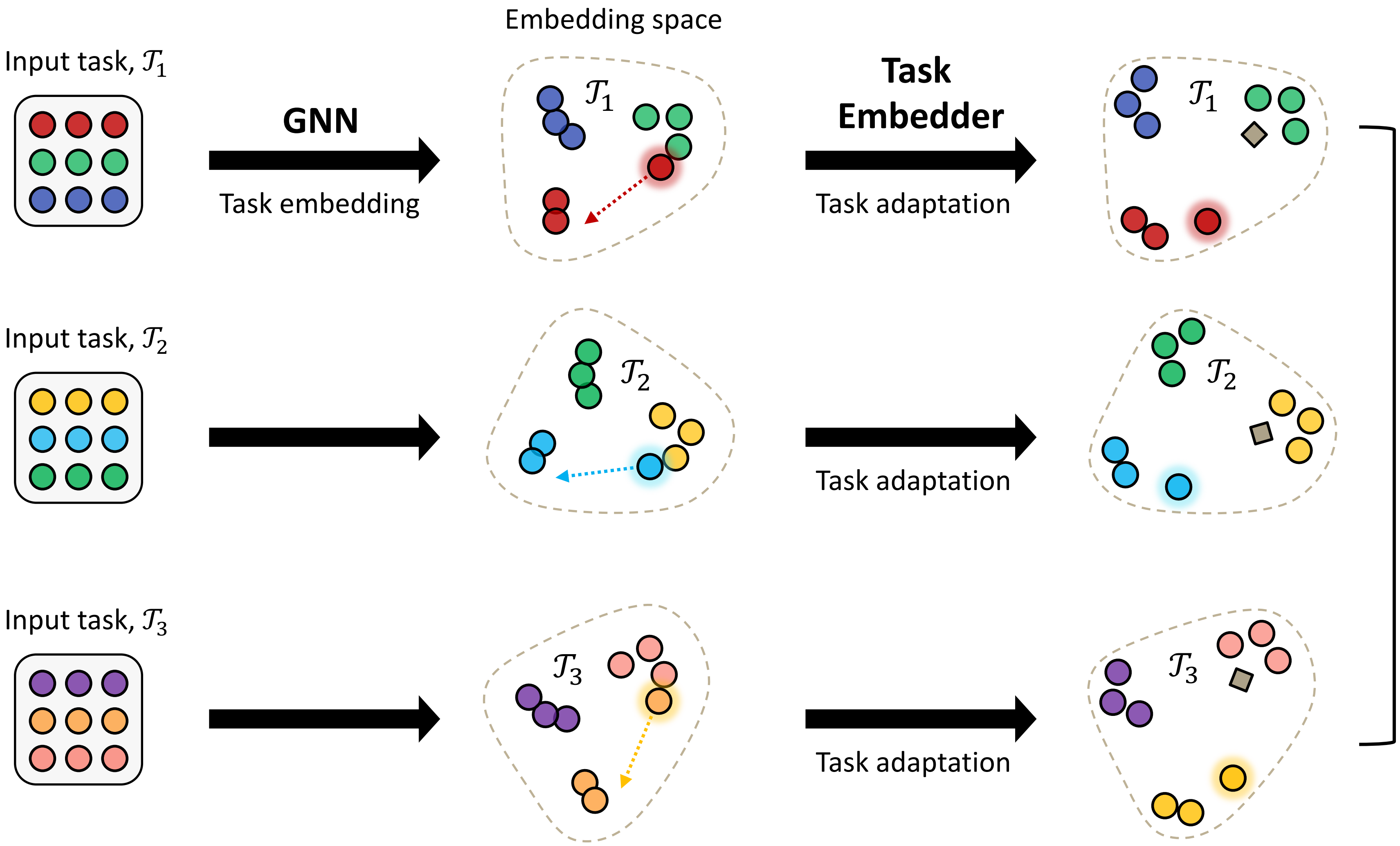
Task-patterns : Relational positions between constituent nodes within the task.

THE GENERAL PROCESS FOR SOLVING THE TASK



Task-patterns : Relational positions between constituent nodes within the task.

THE GENERAL PROCESS FOR SOLVING THE TASK

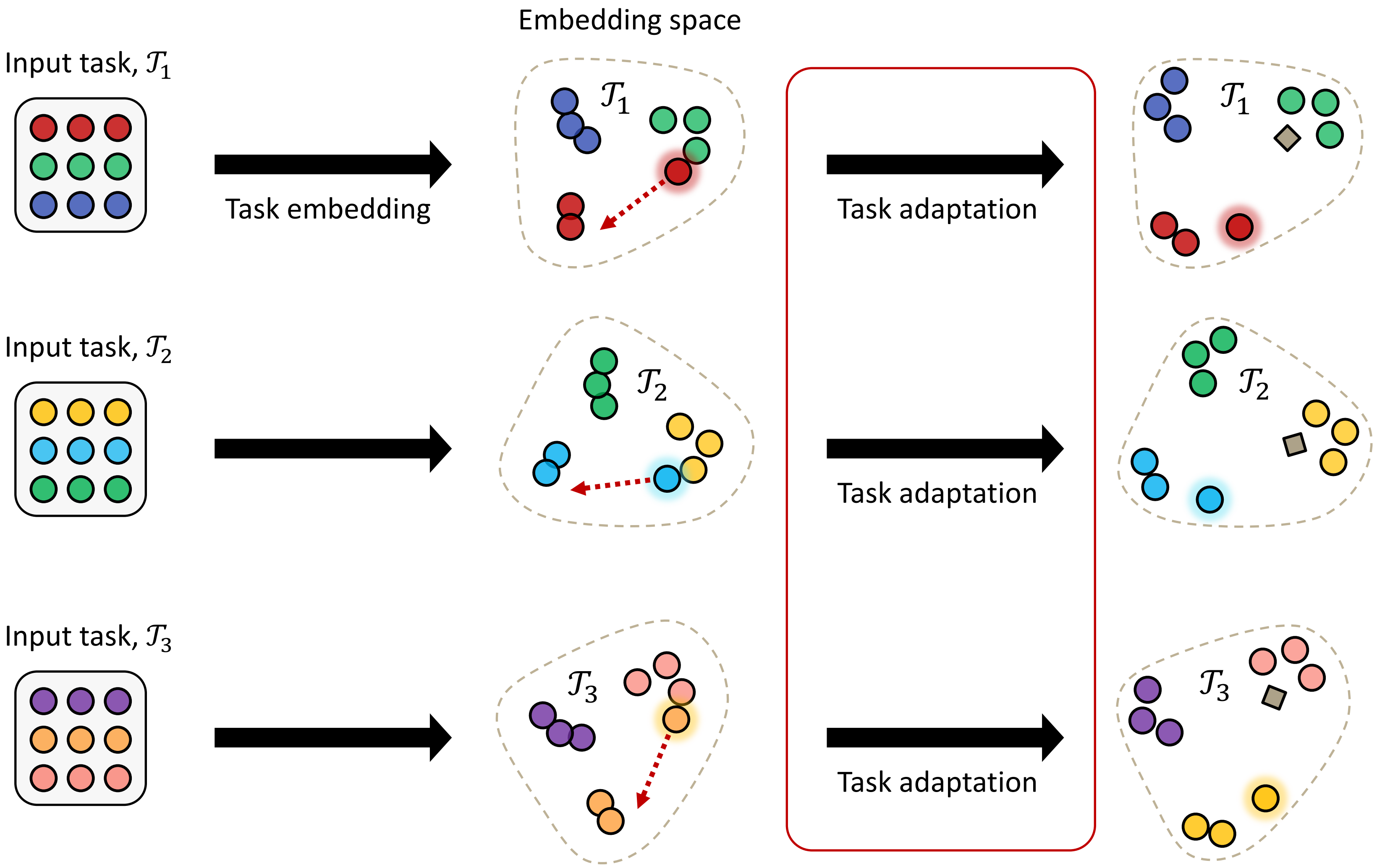


Distinct adaptation knowledge.

↓

Limit generalization power

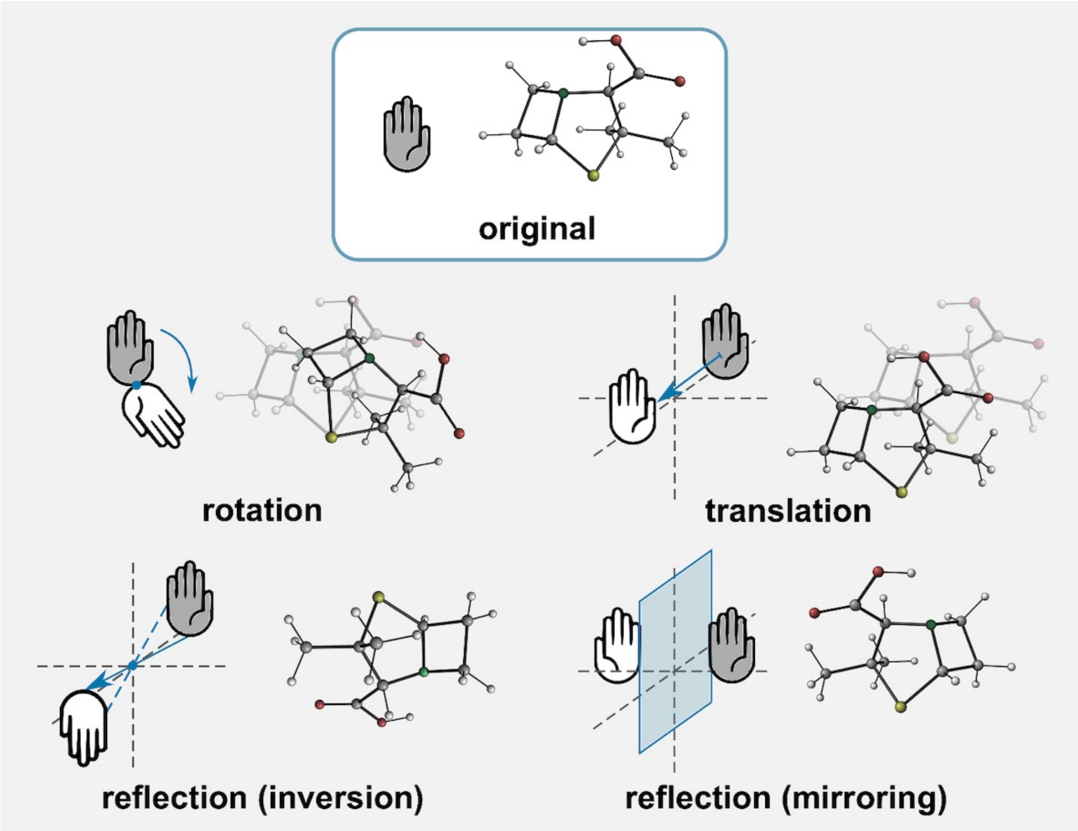
THE GENERAL PROCESS FOR SOLVING THE TASK



Let's share Task-adaptation Strategy! → How?

EQUIVARIANCE

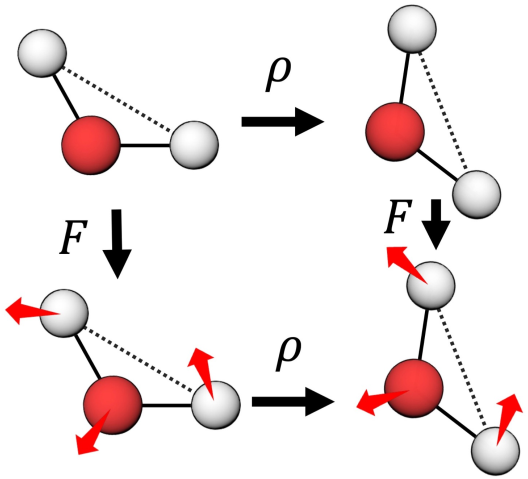
Euclidean Transformations



A function $F: X \rightarrow Y$ is **equivariant** to a transformation ρ
It satisfies:

$$F \circ \rho(x) = \rho \circ F(x)$$

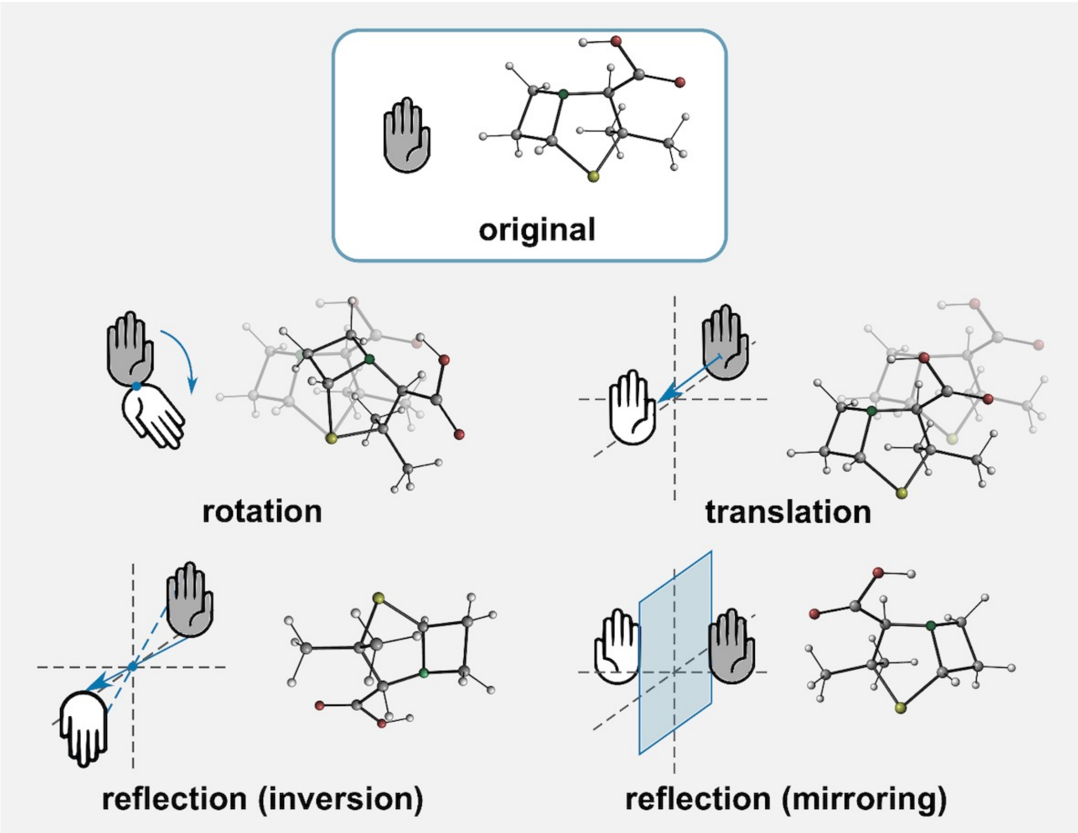
The equation says that **applying ρ on the input** has the **same effect as applying it to the output.**



Atz, Kenneth, Francesca Grisoni, and Gisbert Schneider. "Geometric deep learning on molecular representations." arXiv preprint arXiv:2107.12375 (2021).

EQUIVARIANCE

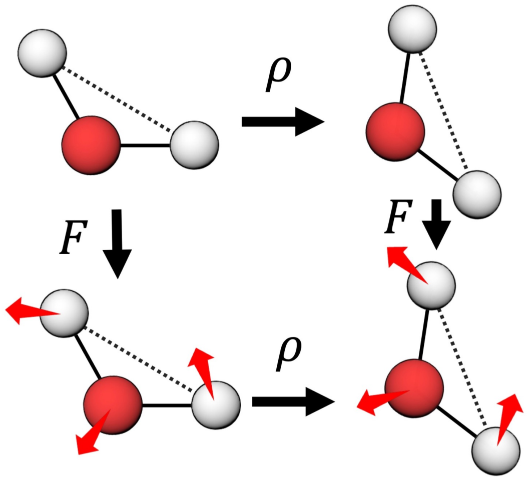
Euclidean Transformations



A function $F: X \rightarrow Y$ is **equivariant** to a transformation ρ
It satisfies:

$$F \circ \rho(x) = \rho \circ F(x)$$

The equation says that **applying ρ on the input** has the **same effect as applying it to the output.**

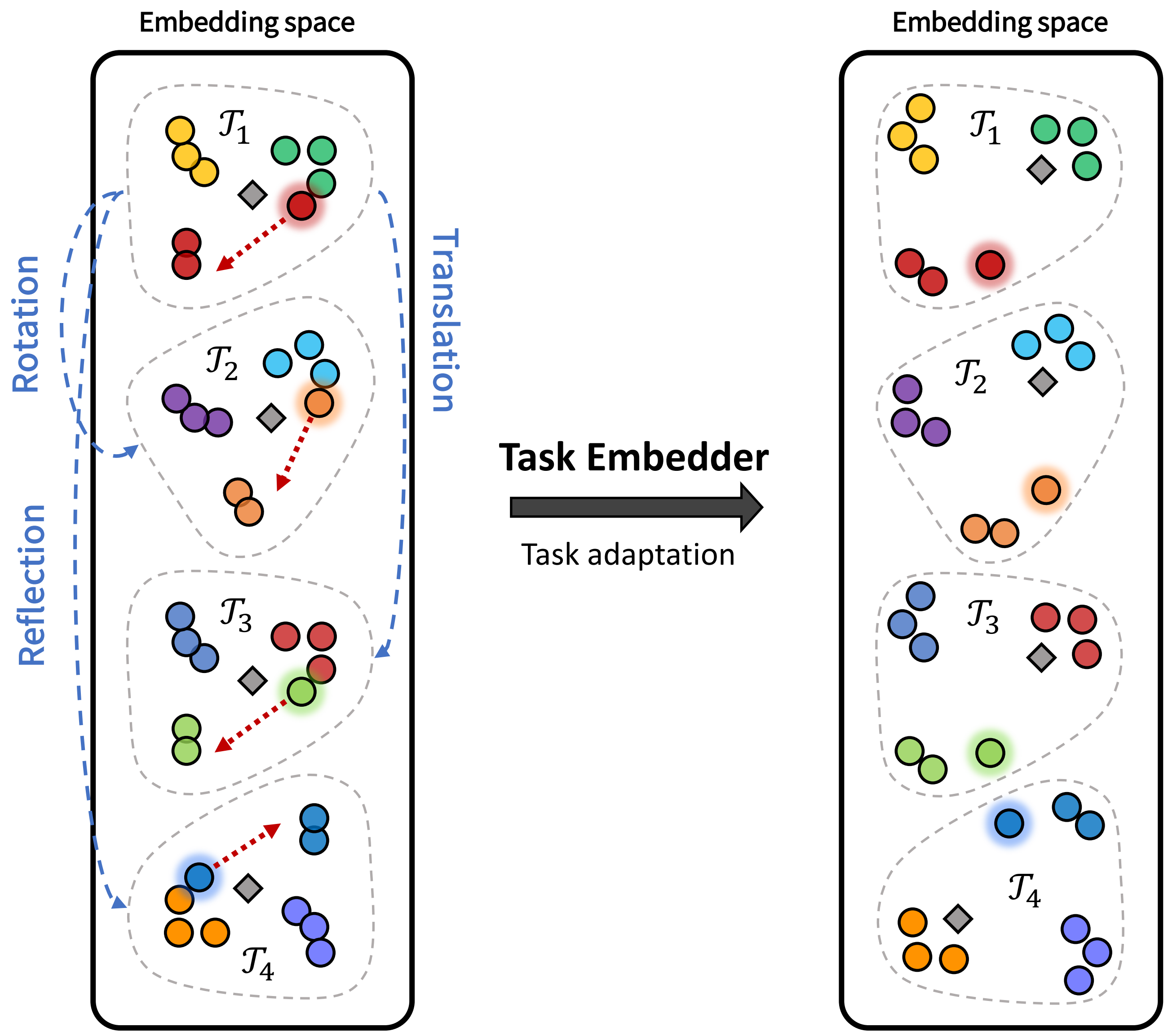


A function $F: X \rightarrow Y$ is **invariant** to a transformation ρ
It satisfies:

$$F \circ \rho(x) = F(x)$$

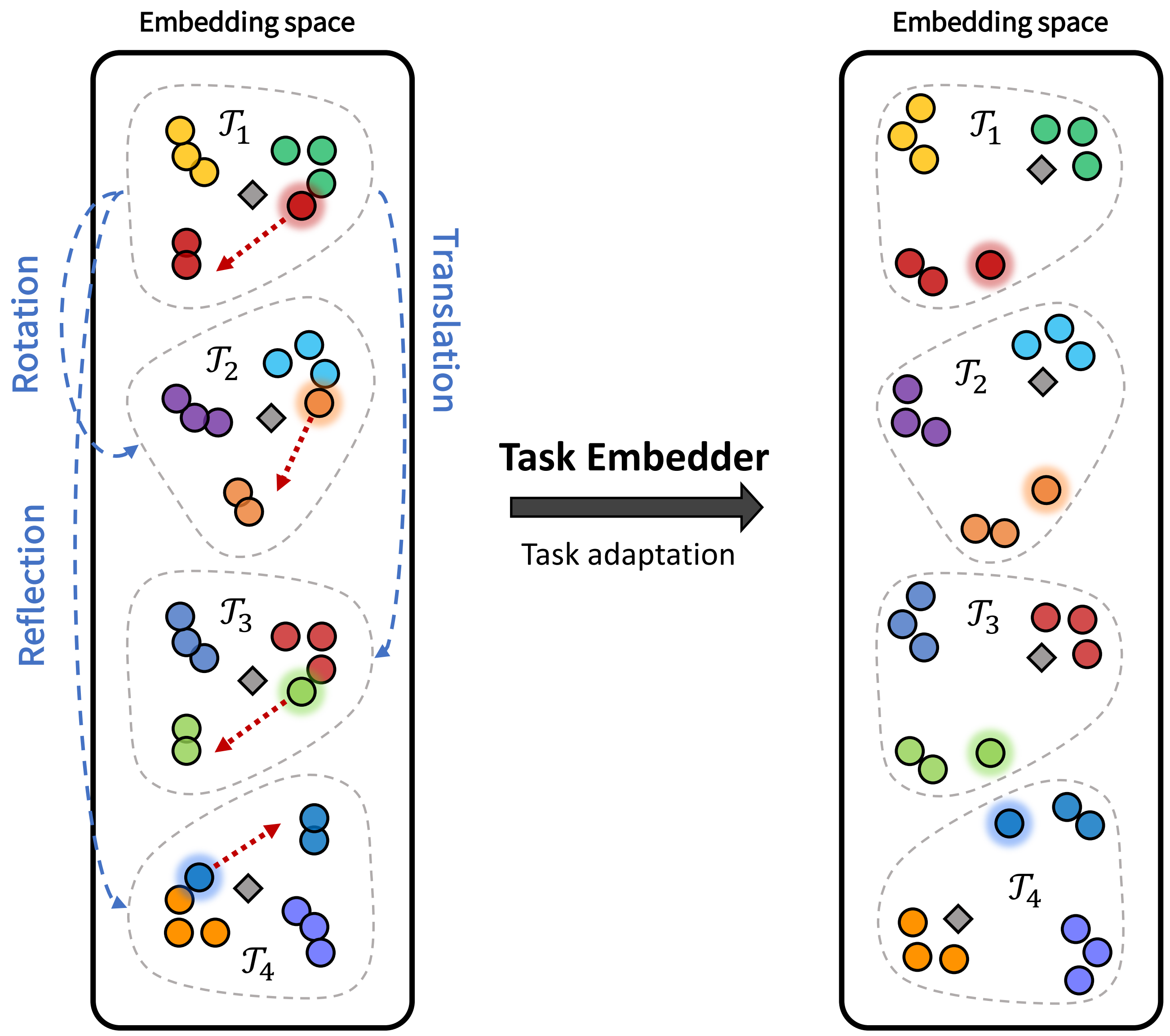
Atz, Kenneth, Francesca Grisoni, and Gisbert Schneider. "Geometric deep learning on molecular representations." arXiv preprint arXiv:2107.12375 (2021).

APPLYING EQUIVARIANCE TO FEW-SHOT LEARNING



Task adaptation strategy exhibits equivariance to transformations of the task embedding.

APPLYING EQUIVARIANCE TO FEW-SHOT LEARNING



Task adaptation strategy exhibits equivariance to transformations of the task embedding.

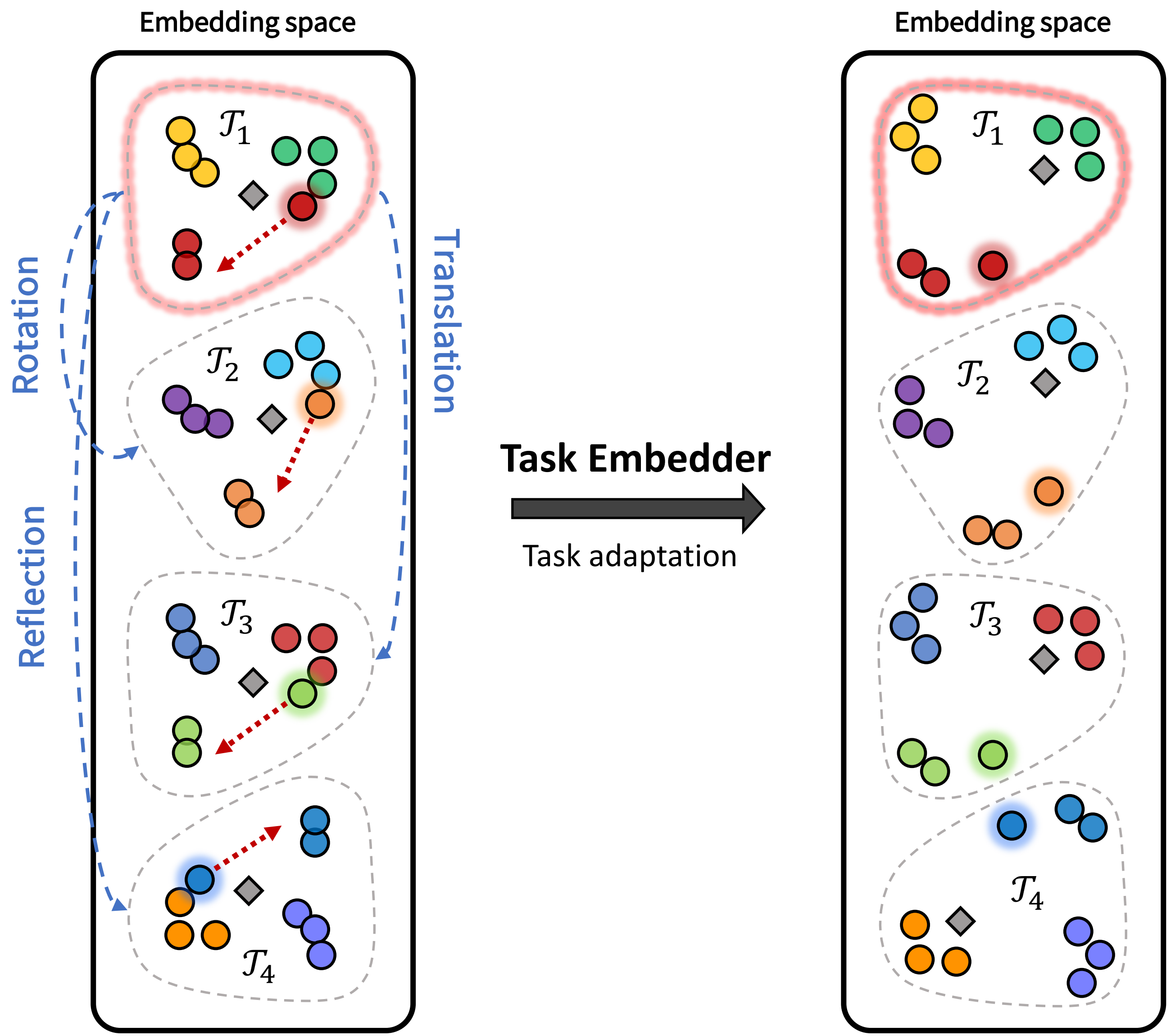
↓

Share adaptation strategies for tasks with same/similar patterns.

→ *Task-Equivariance*

The task embedder is <u>equivariant to Euclidean transformation of embeddings of set of nodes within a task.</u>

APPLYING EQUIVARIANCE TO FEW-SHOT LEARNING



Task adaptation strategy exhibits equivariance to transformations of the task embedding.

↓

Share adaptation strategies for tasks with same/similar patterns. → *Task-Equivariance*

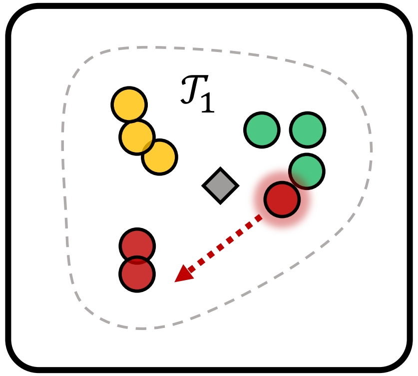
↓

Well-generalized meta-knowledge with low diverse training tasks.

→ Our task embedder can solve $\mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$ if it can handle $\mathcal{T}_1$.
$\mathcal{T}_1$ **is all we need for training data.**

APPLYING EQUIVARIANCE TO FEW-SHOT LEARNING

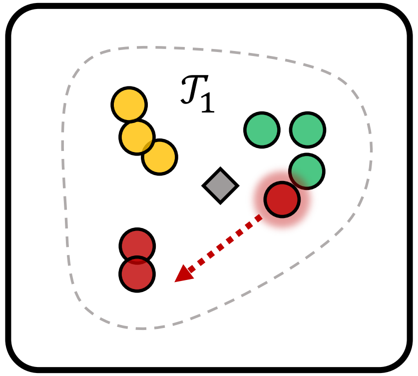
Embedding space



Considering only the relative embedding within a single task **does not provide enough information to distinguish the** **shining red node** from the green nodes.

APPLYING EQUIVARIANCE TO FEW-SHOT LEARNING

Embedding space

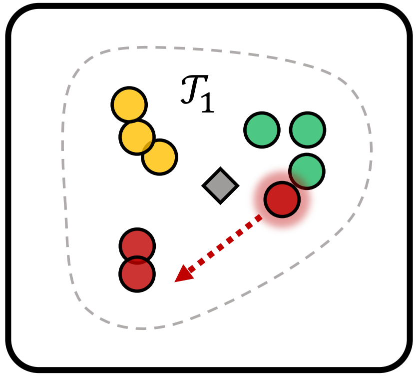


Considering only the relative embedding within a single task **does not provide enough information to distinguish the** **shining red node** from the green nodes.

→ We need the **global information from the entire graph for each node**.

APPLYING EQUIVARIANCE TO FEW-SHOT LEARNING

Embedding space



Considering only the relative embedding within a single task **does not provide enough information to distinguish the shining red node** from the green nodes.

→ We need the **global information from the entire graph for each node**.

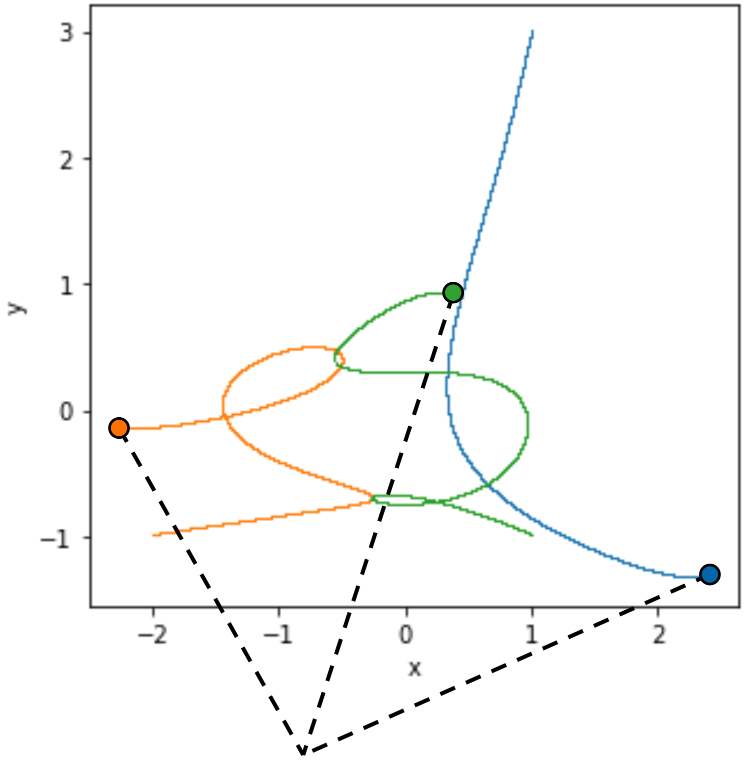
→ ***We generate structural features as global information, which remain constant across all meta-tasks!***

e.g., node2vec, DeepWalk, Shortest Path Distance, Centrality ⋯

→ ***Structural features are constant*** across all meta-tasks!

MIMICKING THE N-BODY PROBLEM

N-body problem

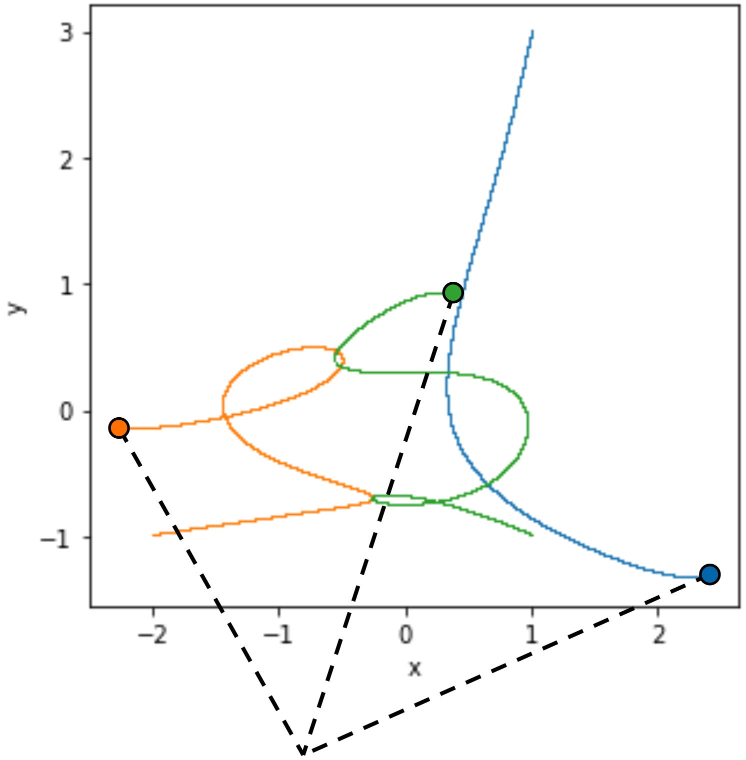


Each instance has its own **1)** ***properties*** (constant) and **2)** ***coordinates*** (relative)

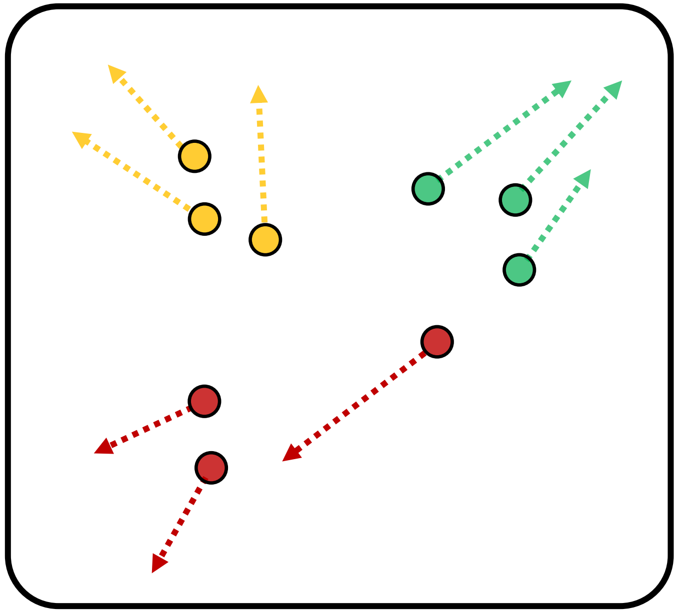
Equivariance is needed.

MIMICKING THE N-BODY PROBLEM

N-body problem



Few-shot Problem



Each instance has its own **1)** ***properties*** (constant) and **2)** ***coordinates*** (relative)

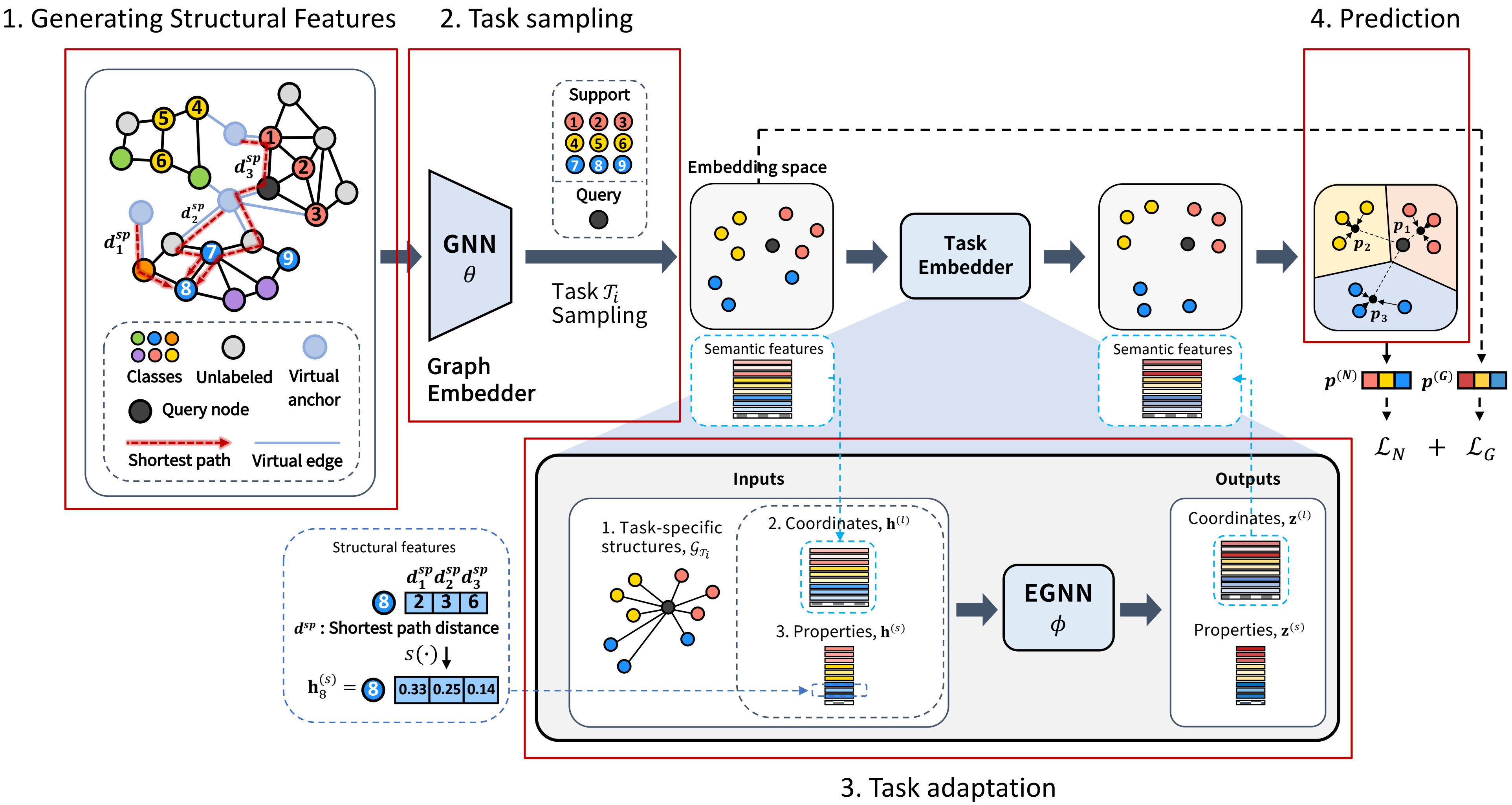
→

1) ***structural features*** (constant)
2) ***embeddings*** (relative)

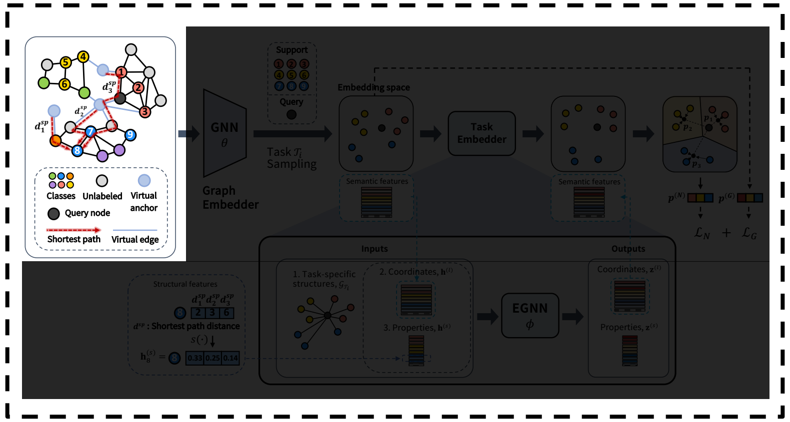
Equivariance is needed.

Equivariance is needed.

METHODOLOGY



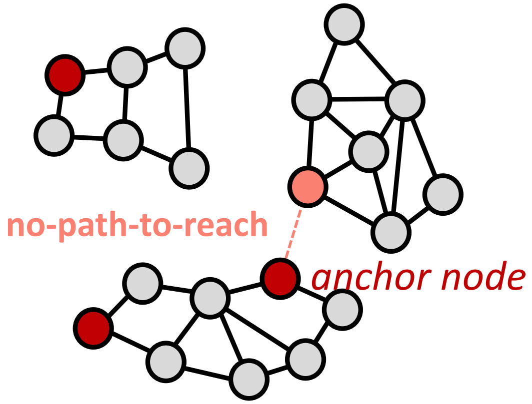
METHODOLOGY



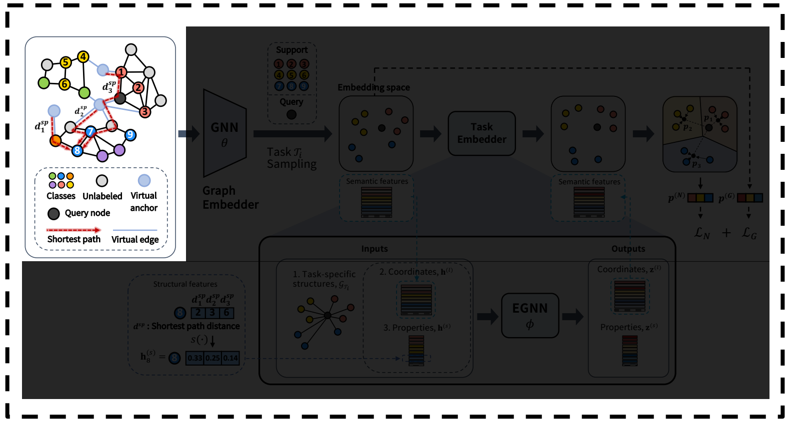
Generating Structural Features (h^s)

Real-world graph datasets tend to consist of multiple connected components.

→ Existing path-based structural features (such as SPD, DeepWalk …) may be hindered by ***no-path-to-reach*** problem.



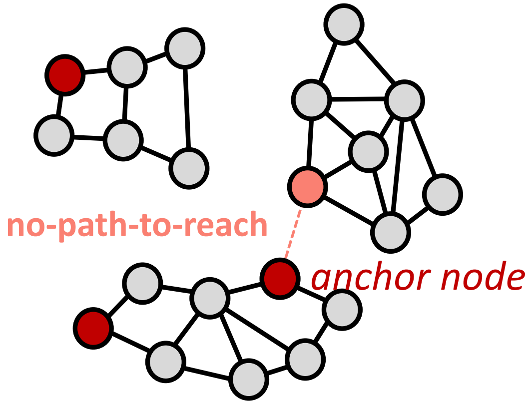
METHODOLOGY



Generating Structural Features (h^s)

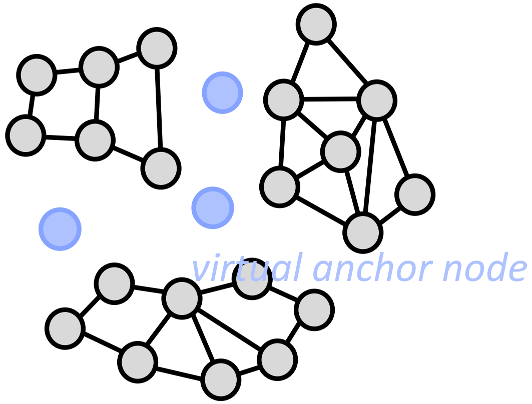
Real-world graph datasets tend to consist of multiple connected components.

→ Existing path-based structural features (such as SPD, DeepWalk …) may be hindered by ***no-path-to-reach*** problem.

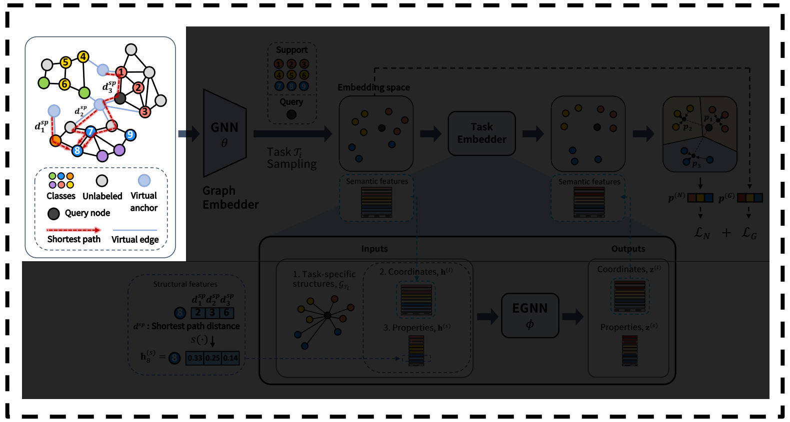


1. Generate k virtual anchor nodes.

$$\mathcal{V}_\alpha = \{v_{\alpha_1}, \ldots, v_{\alpha_k}\}$$



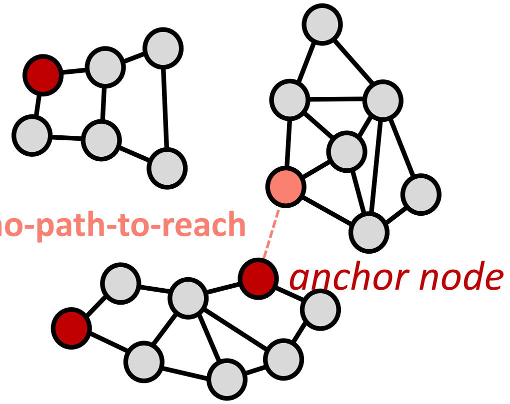
METHODOLOGY

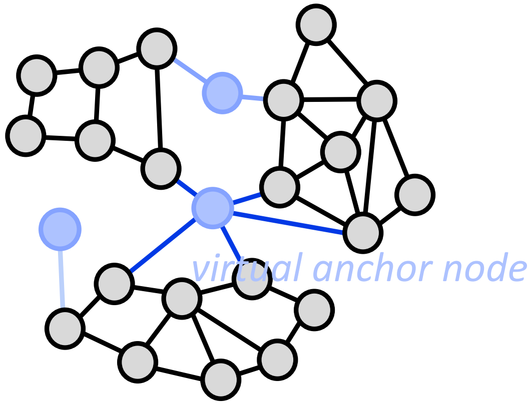


Generating Structural Features (h^s)

Real-world graph datasets tend to consist of multiple connected components.

→ Existing path-based structural features (such as SPD, DeepWalk …) may be hindered by ***no-path-to-reach*** problem.





1. Generate k virtual anchor nodes.

$$\mathcal{V}_\alpha = \{v_{\alpha_1}, \ldots, v_{\alpha_k}\}$$

2. Vary the degrees of connectivity for each virtual anchor node.

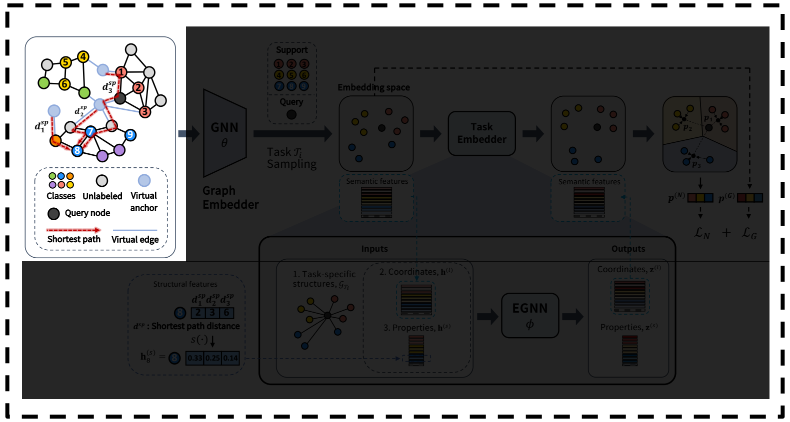
   a) **High degrees**

   → alleviate the no-path-to-reach problem

   b) **Low degrees**

   → has high certainty of structural information.

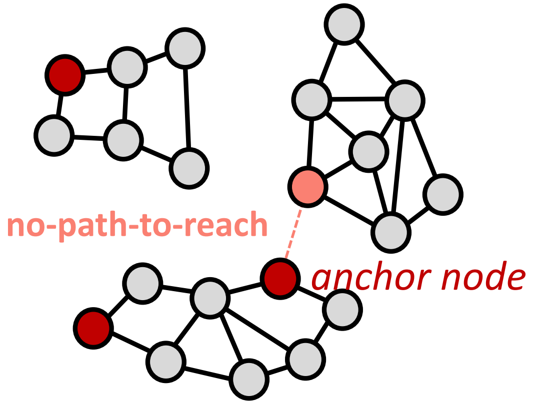
METHODOLOGY

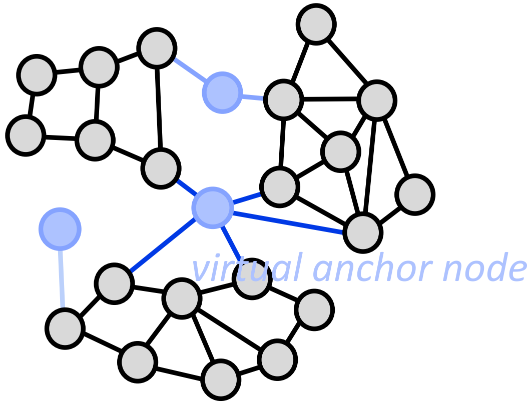


Generating Structural Features (h^s)

Real-world graph datasets tend to consist of multiple connected components.

→ Existing path-based structural features (such as SPD, DeepWalk …) may be hindered by ***no-path-to-reach*** problem.





1. Generate k virtual anchor nodes.

$$\mathcal{V}_\alpha = \{v_{\alpha_1}, \ldots, v_{\alpha_k}\}$$

2. Vary the degrees of connectivity for each virtual anchor node.

   a) **High degrees**

   → alleviate the no-path-to-reach problem

   b) **Low degrees**

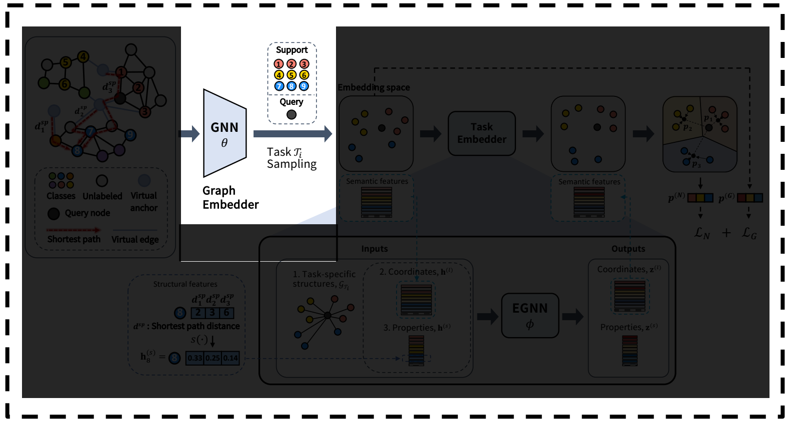
   → has high certainty of structural information.

3. Generate structural features based on the SPD from each k virtual node.

$$\mathbf{H}_v^{(s)} = (s(v, v_{\alpha_1}), s(v, v_{\alpha_2}), \ldots, s(v, v_{\alpha_k}))$$

where $s(v,u) = 1/(d^{sp}(v,u)+1)$ and $d^{sp}(u,v)$ is the SPD between node v and u

METHODOLOGY

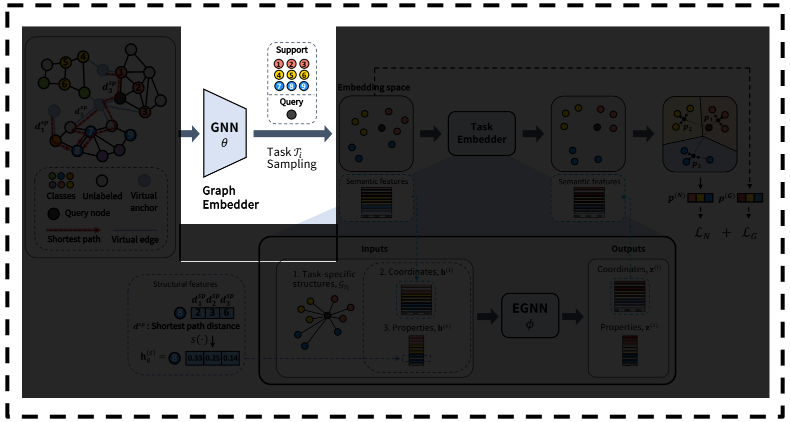


Generating Semantic Features ($\boldsymbol{h^l}$)

In order to reflect the semantic context of the entire graph, we employ GCNs as a graph embedder to obtain the semantic feature $\mathbf{H}^{(l)}$

$$\mathbf{H}^{(l)} = \mathrm{GNN}_{\theta}(\mathbf{X}, \mathbf{A})$$

METHODOLOGY



Generating Semantic Features (h^l)

In order to reflect the semantic context of the entire graph, we employ GCNs as a graph embedder to obtain the semantic feature $\mathbf{H}^{(l)}$

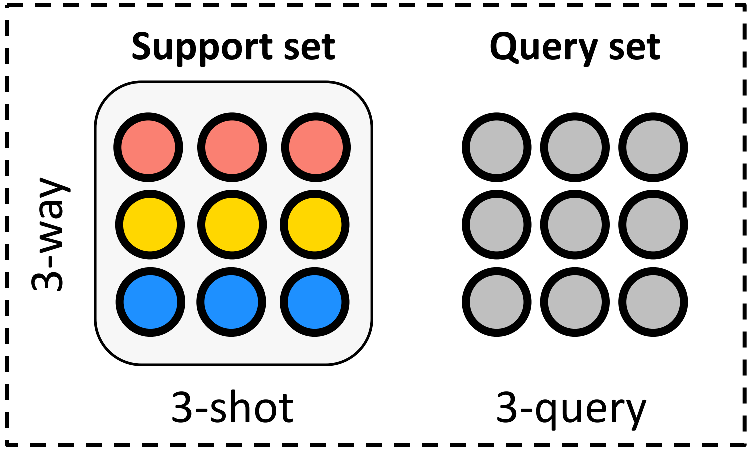
$$\mathbf{H}^{(l)} = \mathrm{GNN}_{\theta}(\mathbf{X},\mathbf{A})$$

Task Sampling

In the case of N-way K-shot, 1) K support nodes 2) M query nodes are samples for each class.

→ $N \times (N + M)$ nodes for each task.

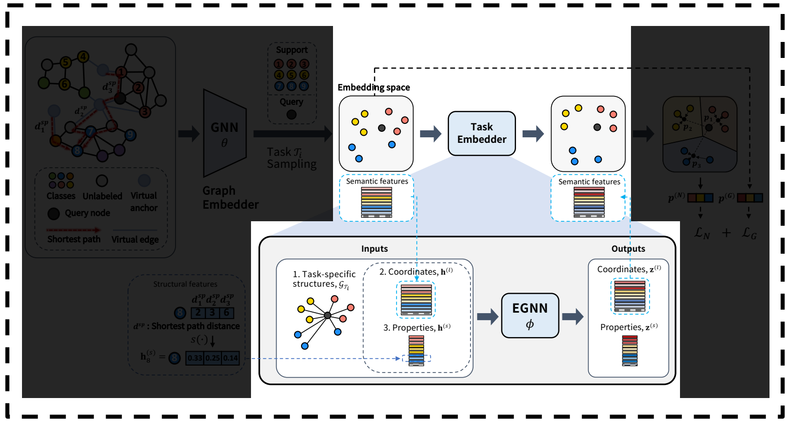
e.g., 3-way 3-shot 3-query task

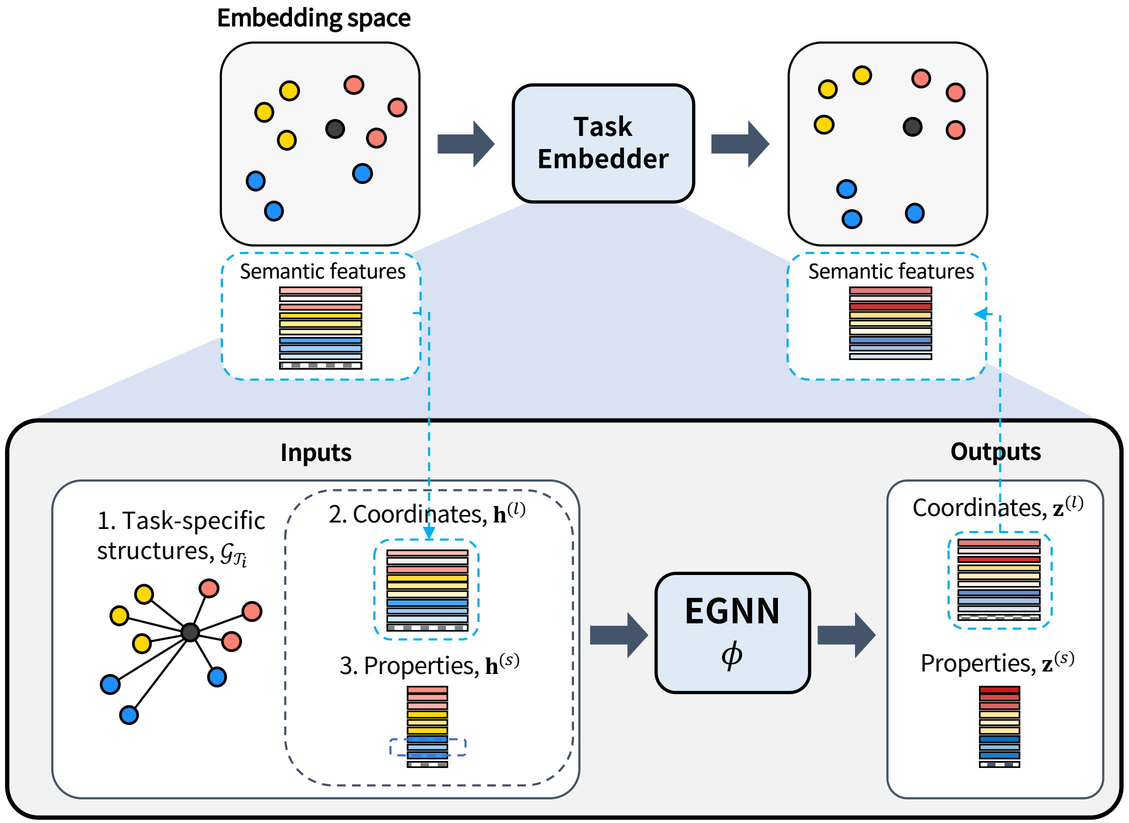


METHODOLOGY

Task Adaptation

Utilizing the Equivariant Graph Neural Networks (EGNN*), the task embedder plays adaptation to the given task.





In order to capture the relations between nodes within the t ask, we use following as inputs :

1. Task-specific graph structures, $\mathcal{G}_{\mathcal{T}_i}$

2. Coordinates of each node in the embedding space.
= Semantic features, $\mathbf{h}^{(l)}$

3. Constant properties of each node across all tasks.
= Structural features, $\mathbf{h}^{(s)}$

* Satorras, Vıctor Garcia, Emiel Hoogeboom, and Max Welling. "E(n) equivariant graph neural networks." *International conference on machine learning*. PMLR, 2021.

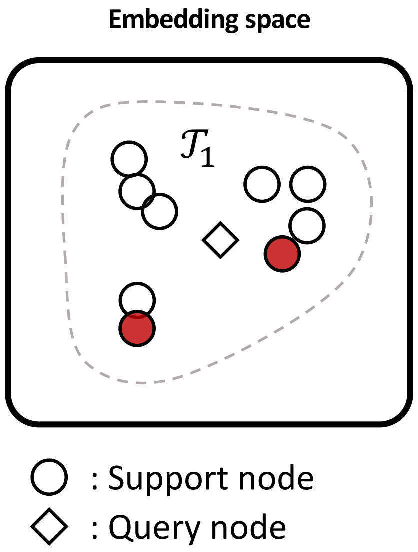
METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

where λ : the index of the layer, $\phi_m : \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.



METHODOLOGY

Task Adaptation

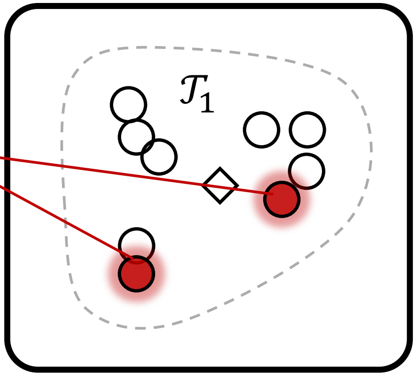
1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\boxed{\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

where λ : the index of the layer, $\phi_m: \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

Constant properties of each node.

Embedding space



: Support node

: Query node

METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

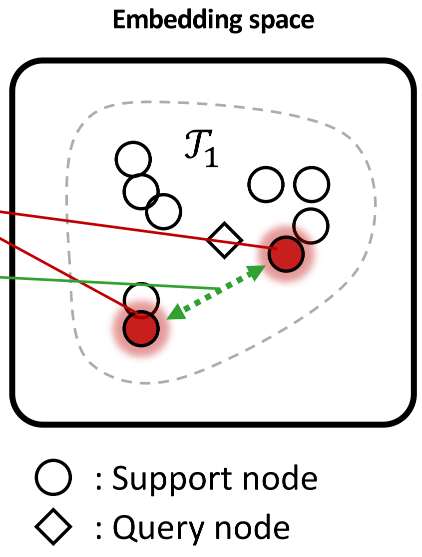
$$\mathbf{m}_{ij} = \phi_m\left(\boxed{\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}}, \boxed{\|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2}\right)$$

Constant properties of each node.

Relative distance between two nodes.

where λ : the index of the lay → **inv.** ${}^{d_s+1} \to \mathbb{R}^{\prime}$ → **inv.**

→ Transformation (i.e., translation, rotation, reflection) **invariant**.



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

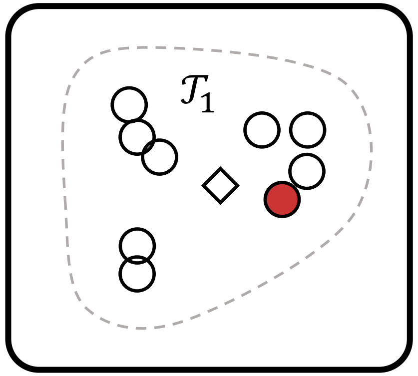
→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m: \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

$$\mathbf{h}_i^{(l),\lambda+1} = \mathbf{h}_i^{(l),\lambda} + \frac{1}{C}\sum_{j\neq i}(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})\phi_l(\mathbf{m}_{ij})$$

where $\phi_l : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

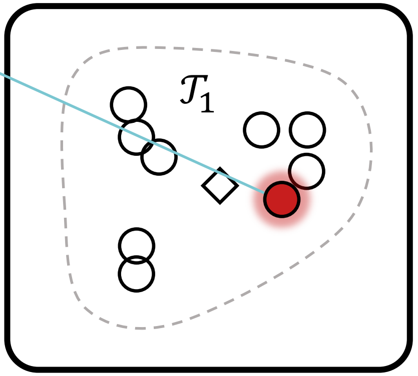
where λ : the index of the layer, $\phi_m: \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

Initial **position** of target node.

$$\mathbf{h}_i^{(l),\lambda+1} = \boxed{\mathbf{h}_i^{(l),\lambda}} + \frac{1}{C}\sum_{j\neq i}(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})\phi_l(\mathbf{m}_{ij})$$

where $\phi_l : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, ϕ_m: $\mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

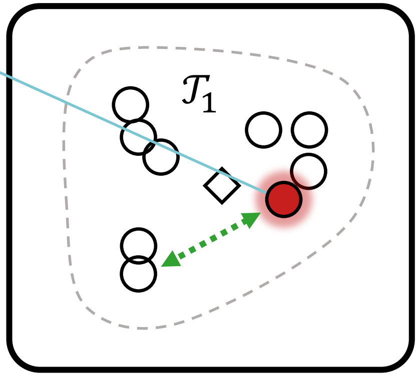
2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

Initial **position** of target node.

$$\mathbf{h}_i^{(l),\lambda+1} = \boxed{\mathbf{h}_i^{(l),\lambda}} + \frac{1}{C}\sum_{j\neq i}\boxed{(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})}\phi_l(\mathbf{m}_{ij})$$

Relative position difference.

where ϕ_l : $\mathbb{R}^{d_l} \rightarrow \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m : \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

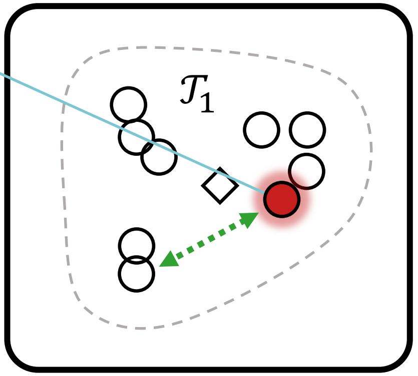
$$\mathbf{h}_i^{(l),\lambda+1} = \boxed{\mathbf{h}_i^{(l),\lambda}} + \frac{1}{C}\sum_{j\neq i} \boxed{(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})} \boxed{\phi_l(\mathbf{m}_{ij})}$$

where $\phi_l : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.

Initial **position** of target node.

Weighted (by message m_{ij})

Relative position difference.



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m : \mathbb{R}^{2d_s+1} \to \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

Initial **position** of target node.

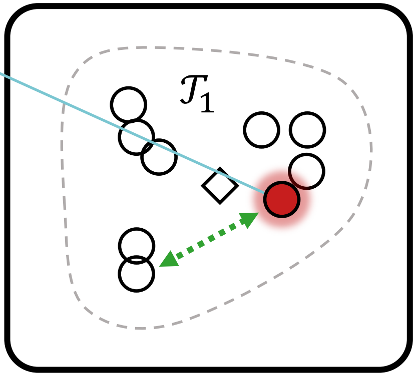
Weighted (by message m_{ij})

Relative position difference.

$$\mathbf{h}_i^{(l),\lambda+1} = \boxed{\mathbf{h}_i^{(l),\lambda}} + \frac{1}{C}\sum_{j \neq i} \boxed{(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})} \boxed{\phi_l(\mathbf{m}_{ij})}$$

where $\phi_l : \mathbb{R}^{d_l} \to \mathbb{R}^1$, → inv. nber of r → equi. → inv. k, excluding node i.

→ Transformation **equivariant**.



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m: \mathbb{R}^{2d_s+1} \to \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

$$\mathbf{h}_i^{(l),\lambda+1} = \mathbf{h}_i^{(l),\lambda} + \frac{1}{C}\sum_{j \neq i}(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})\phi_l(\mathbf{m}_{ij})$$

→ Transformation **equivariant**.

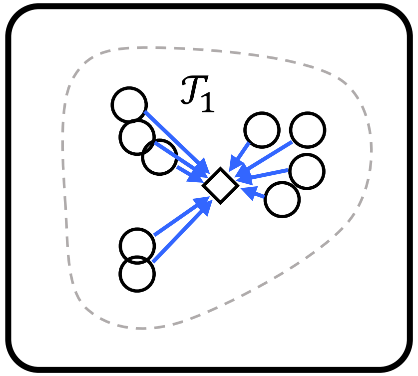
where $\phi_l : \mathbb{R}^{d_l} \to \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.

3. Aggregate messages, then update properties.

$$\mathbf{m}_i = \boxed{\sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}}$$

$$\mathbf{h}_i^{(s),\lambda+1} = \phi_s(\mathbf{h}_i^{(s),\lambda}, \mathbf{m}_i)$$

where $\phi_s : \mathbb{R}^{d_l+d_s} \to \mathbb{R}^{d_s}$



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m : \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

$$\mathbf{h}_i^{(l),\lambda+1} = \mathbf{h}_i^{(l),\lambda} + \frac{1}{C}\sum_{j\neq i}(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})\phi_l(\mathbf{m}_{ij})$$

→ Transformation **equivariant**.

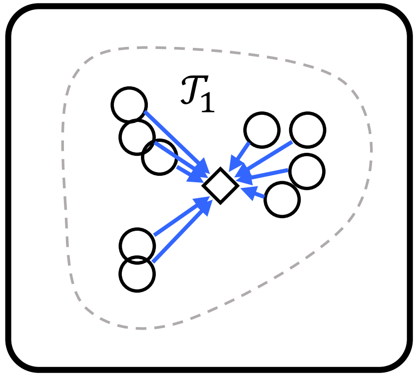
where $\phi_l : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.

3. Aggregate messages, then update properties.

$$\mathbf{m}_i = \boxed{\sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}}$$

$$\mathbf{h}_i^{(s),\lambda+1} = \phi_s(\boxed{\mathbf{h}_i^{(s),\lambda}}, \mathbf{m}_i)$$

where $\phi_s : \mathbb{R}^{d_l+d_s} \rightarrow \mathbb{R}^{d_s}$



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m: \mathbb{R}^{2d_s+1} \to \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

$$\mathbf{h}_i^{(l),\lambda+1} = \mathbf{h}_i^{(l),\lambda} + \frac{1}{C}\sum_{j\neq i}(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})\phi_l(\mathbf{m}_{ij})$$

→ Transformation **equivariant**.

where $\phi_l : \mathbb{R}^{d_l} \to \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.

3. Aggregate messages, then update properties.

$$\mathbf{m}_i = \boxed{\sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}}$$

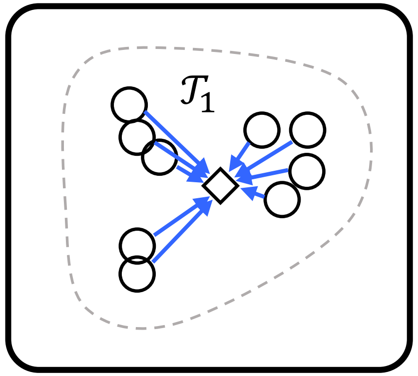
→ **inv.**

$$\mathbf{h}_i^{(s),\lambda+1} = \phi_s(\boxed{\mathbf{h}_i^{(s),\lambda}}, \mathbf{m}_i)$$

→ Transformation **invariant**.

where $\phi_s : \mathbb{R}^{d_l+d_s} \to \mathbb{R}^{d_s}$

→ **inv.**



METHODOLOGY

Task Adaptation

1. Generate a message m_{ij} from node j to i.

$$\mathbf{m}_{ij} = \phi_m(\mathbf{h}_i^{(s),\lambda}, \mathbf{h}_j^{(s),\lambda}, \|\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda}\|^2)$$

→ Transformation (i.e., translation, rotation, reflection) **invariant**.

where λ : the index of the layer, $\phi_m: \mathbb{R}^{2d_s+1} \rightarrow \mathbb{R}^{d_l}$.

2. With the generated messages $\mathbf{m}_{ij}$, update coordinates.

$$\mathbf{h}_i^{(l),\lambda+1} = \mathbf{h}_i^{(l),\lambda} + \frac{1}{C}\sum_{j\neq i}(\mathbf{h}_i^{(l),\lambda} - \mathbf{h}_j^{(l),\lambda})\phi_l(\mathbf{m}_{ij})$$

→ Transformation **equivariant**.

where $\phi_l : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^1$, C : the number of nodes within a meta-task, excluding node i.

3. Aggregate messages, then update properties.

$$\mathbf{m}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij}$$

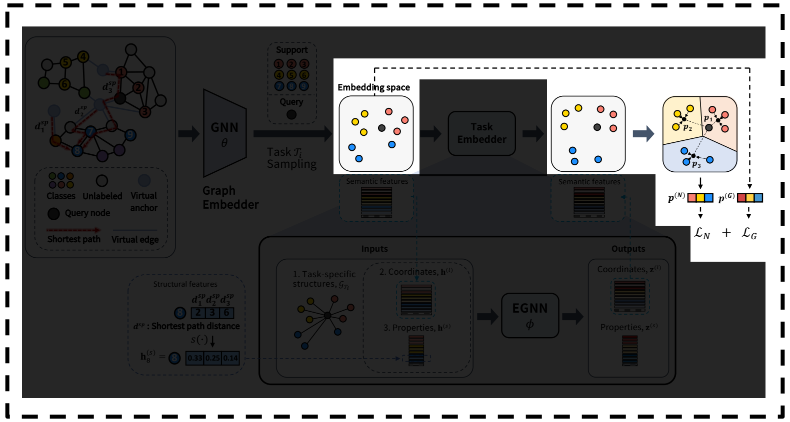
$$\mathbf{h}_i^{(s),\lambda+1} = \phi_s(\mathbf{h}_i^{(s),\lambda}, \mathbf{m}_i)$$

→ Transformation **invariant**.

where $\phi_s : \mathbb{R}^{d_l+d_s} \rightarrow \mathbb{R}^{d_s}$

The task embedder plays an important role where ***adaptation is made equivariantly with respect to the transformation of semantic features.***

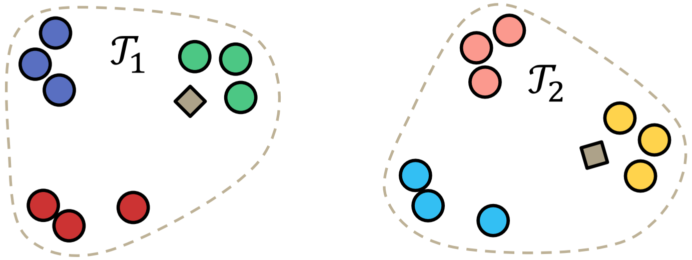
METHODOLOGY



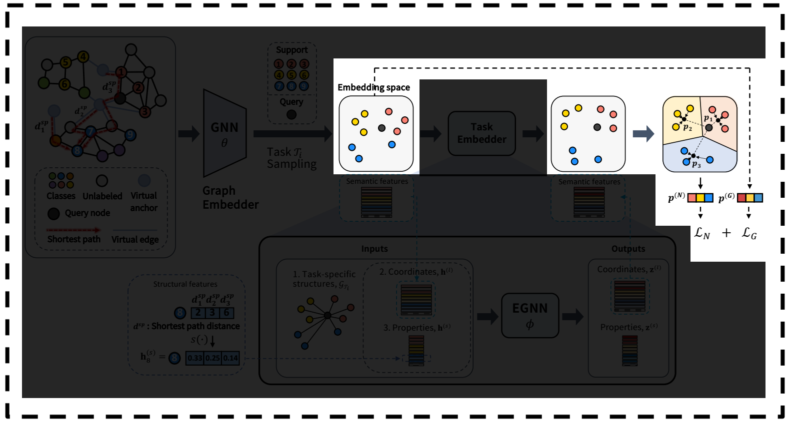
Prediction

The task adaptation strategies have to be equivariant, but we need to provide the same prediction(logits) for different tasks that have same task-patterns.

→ **The metric of prediction should be invariant** to the transformation.



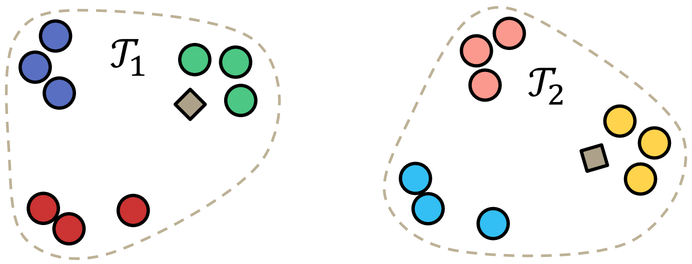
METHODOLOGY



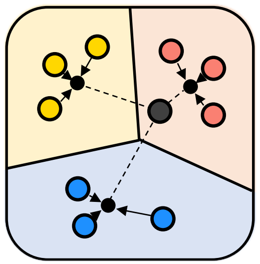
Prediction

The task adaptation strategies have to be equivariant, but we need to provide the same prediction(logits) for different tasks that have same task-patterns.

→ **The metric of prediction should be invariant** to the transformation.



We adopt ProtoNet* based prediction, which are using **squared Euclidean distance, which an invariant metric to transformations.**



$$\mathbf{p}_c^{(N)} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{z}_{c,i}^{(l)}$$

where $\mathbf{z}_{c,i}^{(l)}$: final coordinates of the i-th support nodes, which belongs to class c.

$$p(c|\mathbf{z}_{qry}^{(l)}) = \frac{\exp(-d(\mathbf{z}_{qry}^{(l)}, \mathbf{p}_c^{(N)}))}{\sum_{c'=1}^{N}\exp(-d(\mathbf{z}_{qry}^{(l)}, \mathbf{p}_{c'}^{(N)}))}$$

where $d(\cdot,\cdot)$: squared Euclidean distance.

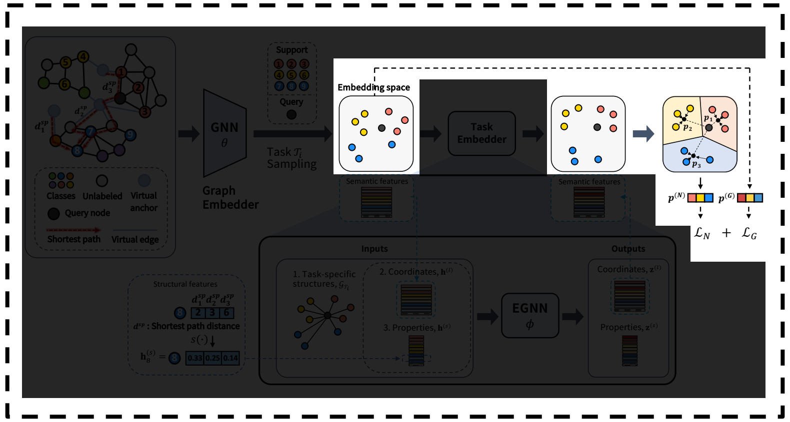
Then we classify the query node by finding the class with the highest probability.

$$\mathcal{L}_N = \sum_{q}^{M}\sum_{c}^{N} -\mathbb{I}(y_q = c)\log(p(c|\mathbf{z}_q^{(l)}))$$

where y_q : ground truth label of the q-th query node, $\mathbb{I}(\cdot)$: indicator function.

* Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. Advances in neural information processing systems, 30.

METHODOLOGY



Prediction

We also calculate the **loss using the semantic features before task adaptation**, which **helps the graph embedder learn more distinguishable semantic features** between the classes.

$$\mathbf{p}_c^{(G)} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{h}_{c,i}^{(l)}$$

where $\mathbf{h}_{c,i}^{(l)}$: final coordinates of the i-th support nodes, which belongs to class c.

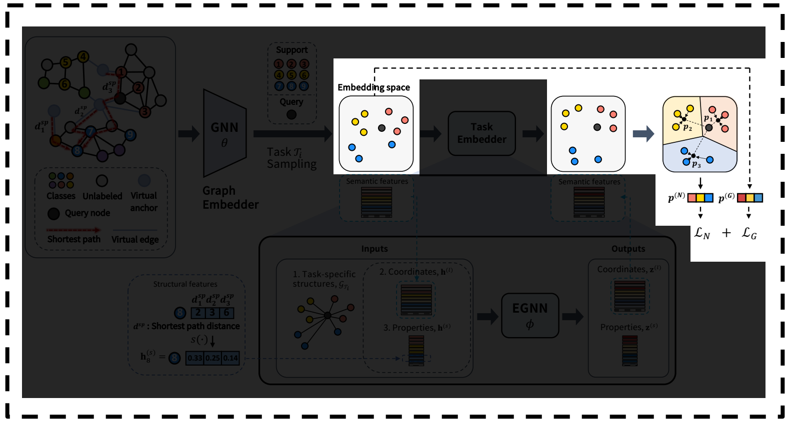
$$p(c|\mathbf{h}_{qry}^{(l)}) = \frac{\exp(-d(\mathbf{h}_{qry}^{(l)}, \mathbf{p}_c^{(G)}))}{\sum_{c'=1}^{N} \exp(-d(\mathbf{h}_{qry}^{(l)}, \mathbf{p}_{c'}^{(G)}))}$$

where $d(\cdot,\cdot)$: squared Euclidean distance.

$$\mathcal{L}_G = \sum_{q}^{M}\sum_{c}^{N} -\mathbb{I}(y_q = c)\log(p(c|\mathbf{h}_q^{(l)}))$$

where y_q : ground truth label of the q-th query node, $\mathbb{I}(\cdot)$: indicator function.

METHODOLOGY



Prediction

We also calculate the **loss using the semantic features before task adaptation**, which **helps the graph embedder learn more distinguishable semantic features** between the classes.

$$\mathbf{p}_c^{(G)} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{h}_{c,i}^{(l)}$$

where $\mathbf{h}_{c,i}^{(l)}$: final coordinates of the i-th support nodes, which belongs to class c.

$$p(c|\mathbf{h}_{qry}^{(l)}) = \frac{\exp(-d(\mathbf{h}_{qry}^{(l)}, \mathbf{p}_c^{(G)}))}{\sum_{c'=1}^{N}\exp(-d(\mathbf{h}_{qry}^{(l)}, \mathbf{p}_{c'}^{(G)}))}$$

where $d(\cdot,\cdot)$: squared Euclidean distance.

$$\mathcal{L}_G = \sum_{q}^{M}\sum_{c}^{N} -\mathbb{I}(y_q = c)\log(p(c|\mathbf{h}_q^{(l)}))$$

where y_q : ground truth label of the q-th query node, $\mathbb{I}(\cdot)$: indicator function.

Final Loss Function

$$\mathcal{L}(\theta, \phi) = \gamma \underbrace{\mathcal{L}_N}_{\text{Task embedder}} + (1-\gamma)\underbrace{\mathcal{L}_G}_{\text{Graph embedder}}$$

where γ: tunable hyperparameter

EXPERIMENTS

Main Results

Dataset	Cora-full						Amazon Clothing					
Method	5way 1shot	5way 3shot	5way 5shot	10way 1shot	10way 3shot	10way 5shot	5way 1shot	5way 3shot	5way 5shot	10way 1shot	10way 3shot	10way 5shot
MAML	24.74 ± 3.20	28.32 ± 1.83	30.13 ± 4.33	10.11 ± 0.49	10.98 ± 1.02	12.89 ± 1.78	45.60 ± 7.16	58.82 ± 5.52	64.88 ± 1.89	29.00 ± 1.86	39.52 ± 2.99	43.98 ± 2.27
ProtoNet	31.47 ± 1.65	39.49 ± 1.46	44.98 ± 1.08	19.75 ± 0.71	28.16 ± 1.73	31.34 ± 0.91	42.37 ± 2.42	57.74 ± 1.09	62.83 ± 3.10	34.51 ± 2.13	49.16 ± 2.72	54.16 ± 1.62
Meta-GNN	51.57 ± 2.83	58.10 ± 2.57	62.66 ± 5.58	29.20 ± 2.36	32.10 ± 4.60	41.36 ± 2.25	70.42 ± 1.66	76.72 ± 2.65	76.27 ± 1.87	51.05 ± 1.53	56.70 ± 2.22	57.54 ± 3.71
G-Meta	45.71 ± 1.97	54.64 ± 2.24	58.68 ± 5.16	32.90 ± 0.84	46.60 ± 0.62	51.58 ± 1.23	61.71 ± 1.67	67.94 ± 1.99	73.28 ± 1.84	50.33 ± 1.62	62.07 ± 1.12	67.23 ± 1.79
GPN	51.09 ± 3.55	63.78 ± 0.66	65.89 ± 2.53	40.24 ± 1.94	50.49 ± 2.34	53.75 ± 2.13	61.39 ± 1.97	73.42 ± 2.77	76.40 ± 2.37	51.32 ± 1.30	64.58 ± 3.04	69.03 ± 0.98
TENT	54.19 ± 2.23	65.20 ± 1.99	68.77 ± 2.42	37.72 ± 2.08	48.76 ± 1.95	53.95 ± 0.81	75.52 ± 1.06	85.21 ± 0.79	87.15 ± 1.13	60.70 ± 1.66	72.44 ± 1.81	77.53 ± 0.76
TEG	**60.27 ± 1.93**	**74.24 ± 1.03**	**76.37 ± 1.92**	**45.26 ± 1.03**	**60.00 ± 1.16**	**64.56 ± 1.04**	**80.77 ± 3.32**	**90.14 ± 0.97**	**90.18 ± 0.95**	**69.12 ± 1.75**	**79.42 ± 1.34**	**83.27 ± 0.81**

Dataset	Amazon Electronics						DBLP					
Method	5way 1shot	5way 3shot	5way 5shot	10way 1shot	10way 3shot	10way 5shot	5way 1shot	5way 3shot	5way 5shot	10way 1shot	10way 3shot	10way 5shot
MAML	41.57 ± 6.32	54.88 ± 2.84	62.90 ± 3.81	28.75 ± 1.70	40.75 ± 3.20	41.98 ± 5.38	31.57 ± 3.57	43.52 ± 5.50	51.09 ± 5.68	16.05 ± 2.27	25.64 ± 2.24	25.66 ± 5.12
ProtoNet	42.38 ± 1.62	52.94 ± 1.31	59.34 ± 2.06	32.05 ± 3.23	43.26 ± 1.72	49.49 ± 3.01	35.12 ± 0.95	49.27 ± 2.70	53.65 ± 1.62	24.30 ± 0.76	39.42 ± 2.03	44.06 ± 1.57
Meta-GNN	57.23 ± 1.54	66.19 ± 2.40	70.08 ± 2.14	41.22 ± 2.85	48.94 ± 1.87	53.55 ± 1.51	63.07 ± 1.49	71.76 ± 2.17	74.70 ± 2.09	45.74 ± 1.68	53.34 ± 2.58	56.14 ± 0.88
G-Meta	47.14 ± 1.24	59.75 ± 1.29	62.06 ± 1.98	41.22 ± 1.86	48.64 ± 1.80	54.49 ± 2.37	57.98 ± 1.98	68.19 ± 1.40	73.11 ± 0.81	47.38 ± 2.72	60.83 ± 1.35	66.12 ± 1.79
GPN	48.32 ± 3.40	63.41 ± 1.54	68.48 ± 2.38	40.34 ± 1.86	53.82 ± 1.24	59.58 ± 1.39	60.43 ± 3.06	68.90 ± 0.54	74.03 ± 1.77	49.73 ± 1.64	62.34 ± 1.67	64.48 ± 2.43
TENT	69.26 ± 1.32	79.12 ± 0.97	81.65 ± 1.31	56.93 ± 1.65	68.56 ± 2.05	72.72 ± 0.78	72.19 ± 1.92	81.84 ± 1.82	82.76 ± 1.29	58.40 ± 1.41	68.55 ± 1.38	72.47 ± 1.27
TEG	**73.78 ± 0.93**	**84.78 ± 1.52**	**87.17 ± 1.15**	**61.34 ± 1.58**	**76.48 ± 1.36**	**79.63 ± 0.73**	**74.32 ± 1.66**	**83.10 ± 2.01**	**83.33 ± 1.22**	**61.81 ± 2.02**	**71.25 ± 1.23**	**74.50 ± 1.49**

*In a traditional few-shot learning settings (i.e., **using sufficient training meta-tasks**), TEG outperforms all the baselines.*

EXPERIMENTS

Impact of Diversity of Meta-Train Tasks

Dataset	Amazon Electronics						Amazon Clothing					
Setting	5way 5shot			10way 5shot			5way 5shot			10way 5shot		
Class/label Avail.	50%/10%	30%/2%	10%/1%	50%/10%	30%/2%	10%/1%	50%/10%	30%/2%	10%/1%	50%/10%	30%/2%	10%/1%
MAML	58.50	55.10	52.00	44.31	40.48	34.04	58.62	53.30	50.16	38.22	33.70	34.46
ProtoNet	54.93	54.86	47.15	47.75	42.80	33.93	57.78	51.89	46.74	43.21	37.22	37.02
Meta-GNN	68.10	62.45	56.24	47.70	41.23	33.86	75.28	73.73	66.29	54.18	50.83	45.70
G-Meta	58.62	53.30	50.16	38.22	33.70	34.46	58.50	55.10	52.00	44.31	40.48	34.04
GPN	69.68	62.14	55.33	58.66	51.06	45.51	73.06	71.06	70.66	65.25	61.24	60.59
TENT	74.90	70.66	56.16	64.43	60.11	48.46	80.40	77.38	65.15	68.91	63.16	60.46
TEG	**83.26**	**81.84**	**76.77**	**75.37**	**72.61**	**68.98**	**88.26**	**86.72**	**82.54**	**80.88**	**78.76**	**78.41**
Rel Improv.	11.2%	15.8%	36.5%	17.0%	20.8%	42.3%	9.8%	12.1%	16.8%	17.4%	24.7%	29.4%

Our model achieves **further performance improvements** compared to the baseline methods **as the diversity of tasks decreases.**

*TEG outperforms other models when faced with limited meta-training tasks and **has a strong ability to adapt to new tasks with minimal training data**, which is common in real-world scenarios.*

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model perfor mance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

1. Train models with meta-train tasks. → 2. Transform the meta-train tasks. → 3. Re-evaluate the models!

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model perfor mance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

1. **Train models with meta-train tasks.** → 2. Transform the meta-train tasks. → 3. Re-evaluate the models!

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model perfor mance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

1. Train models with meta-train tasks. → 2. **Transform the meta-train tasks.** → 3. Re-evaluate the models!

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model perfor mance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

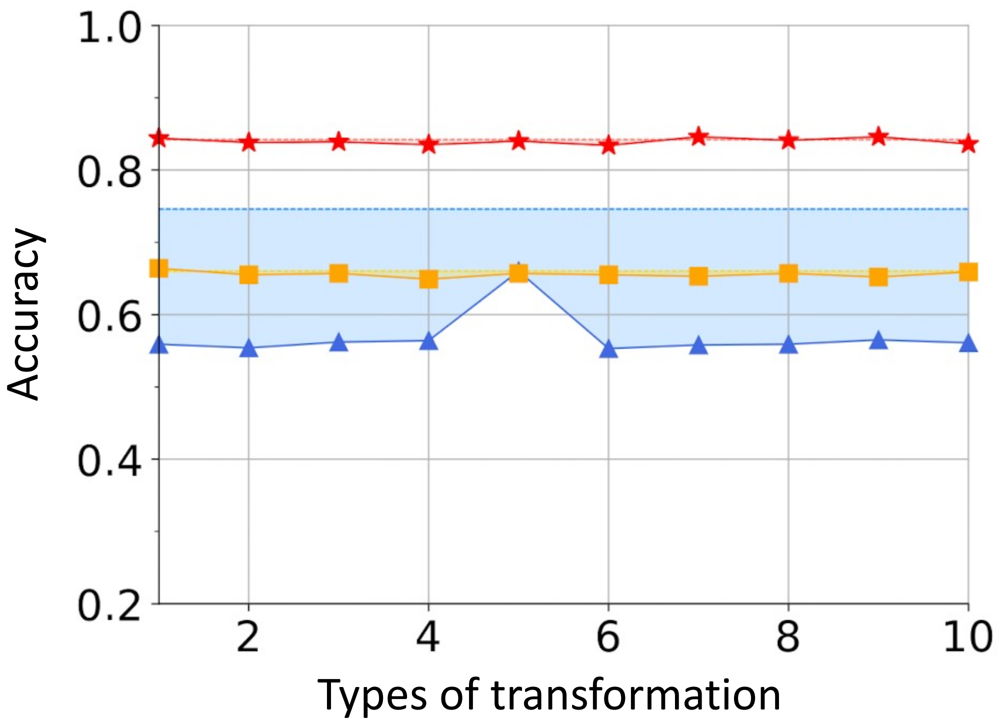
1. Train models with meta-train tasks. → 2. Transform the meta-train tasks. → 3**. Re-evaluate the models!**

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model perfor mance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

1. Train models with meta-train tasks. → 2. Transform the meta-train tasks. → 3. Re-evaluate the models!



T_{ref} : original meta-train tasks.

T_{trans} : transformed meta-train tasks.

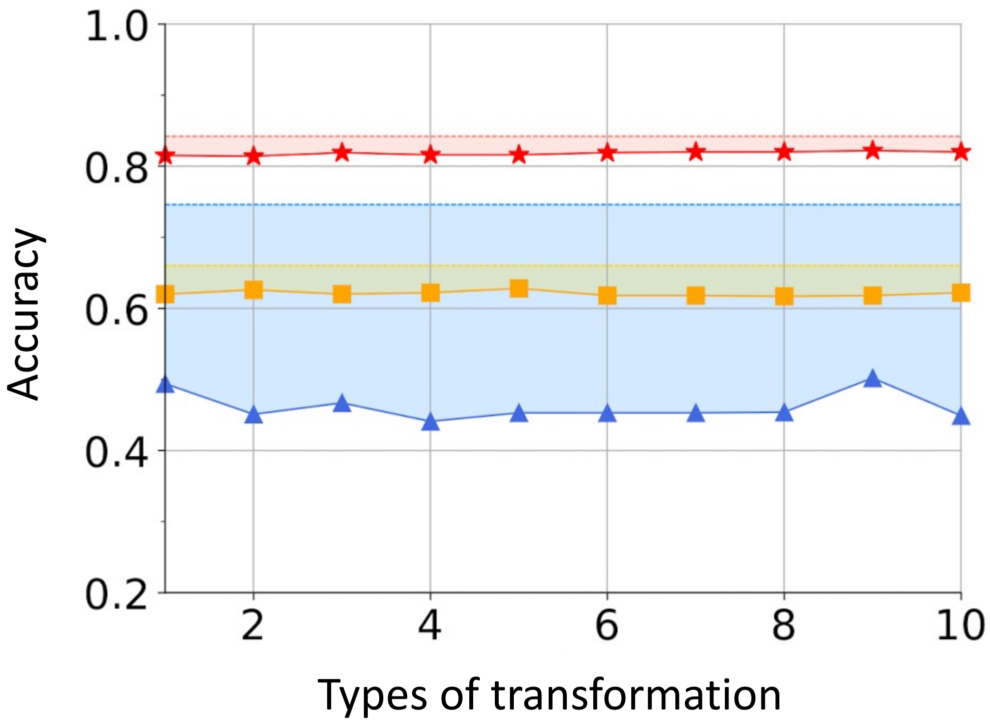
(a) Transformation only → Evaluation for Tasks with **same** patterns

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model performance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

1. Train models with meta-train tasks. → 2. Transform the meta-train tasks. → 3. Re-evaluate the models!



T_{ref} : original meta-train tasks.
T_{trans} : transformed meta-train tasks.

TEG(ref) TEG(trans)
TENT(ref) TENT(trans)
GPN(ref) GPN(trans)

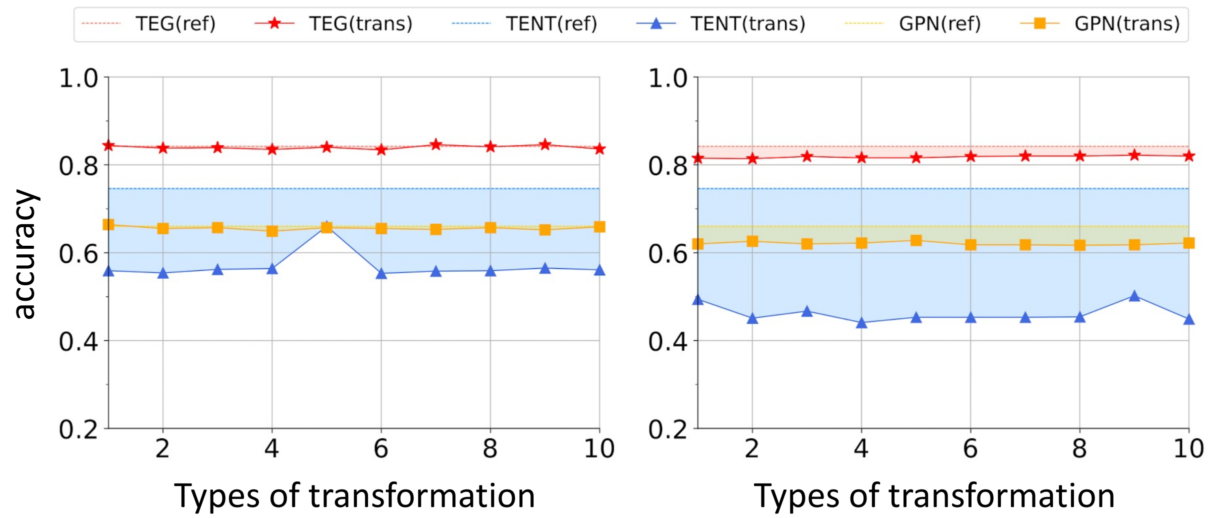
(b) Transformation with noises → Evaluation for Tasks with **similar** patterns

EXPERIMENTS

Effectiveness of *Task-Equivariance*

In order to verify the **generalization ability of TEG achieved by the task-equivariance**, we evaluate the model perfor mance on a set of meta-tasks generated by **transforming the meta-train tasks set.**

1. Train models with meta-train tasks. → 2. Transform the meta-train tasks. → 3. Re-evaluate the models!



T_{ref} : original meta-train tasks.

T_{trans} : transformed meta-train tasks.

(a) Transformation only — Tasks with **same** patterns

(b) Transformation with noises — Tasks with **similar** patterns

Task-equivariance *enables the model to* ***acquire highly transferable meta-knowledge*** *that can be applied to new tasks with both same and similar task- patterns.*

CONCLUSION

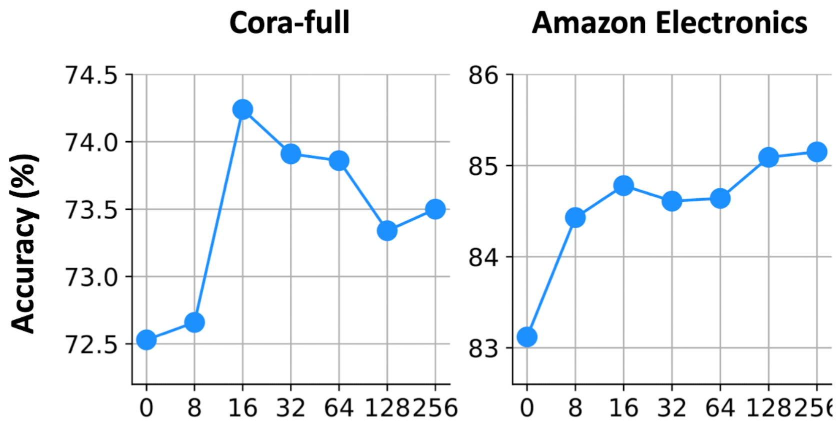
- In meta-learning based few-shot learning, **having a sufficient number of training meta-tasks is crucial**.
- However, **obtaining diverse training meta-tasks is challenging** in real-world scenarios due to the high cost of labeling.
- To address this, TEG learns **highly transferable task-adaptation strategies even from limited training meta-tasks** with low diversity.
- We **incorporate equivariance into few-shot learning** to maximize generalization with the limited tasks.

THANK YOU

APPENDIX

Table 5: Effect of using virtual anchor node for alleviating no-path-to-reach problem. AC and AE denotes "Amazon Clothing" and "Amazon Electronics", respectively.

		with virtual anchor nodes		w.o. virtual anchor nodes	
	Dimension	# Zero value	Zero ratio	# Zero value	Zero ratio
Corafull	19,793 × 16	544	0.002	15,888	0.050
AC	24,919 × 16	1,280	0.003	66,662	0.167
AE	42,318 × 16	9,472	0.014	666,935	0.985
DBLP	40,672 × 16	0	0.000	352	0.001



(a) Number of virtual anchor node k

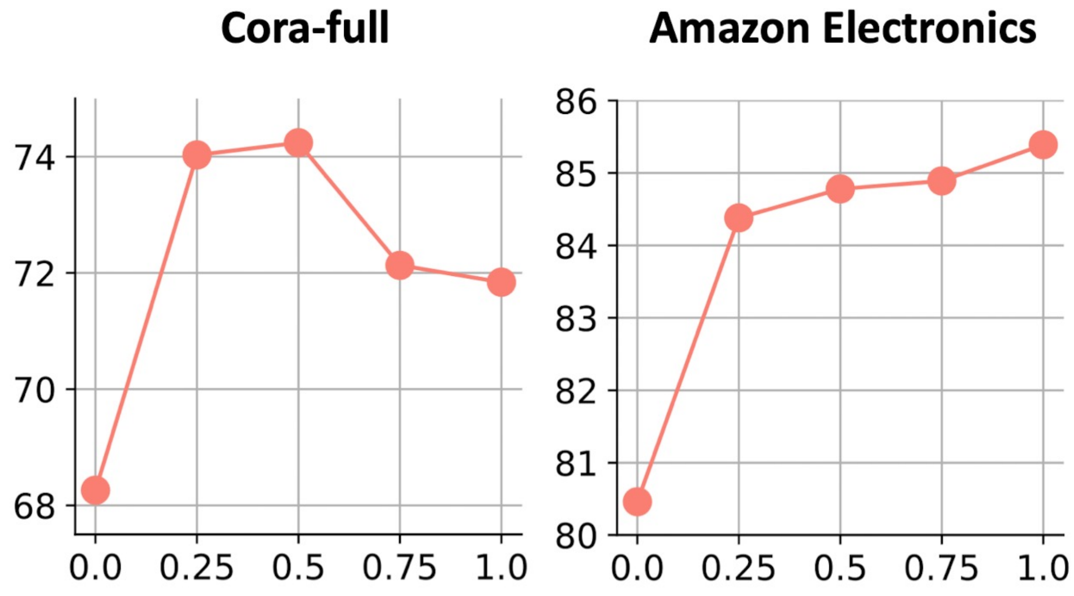
APPENDIX

Final Loss Function

$$\mathcal{L}(\theta, \phi) = \gamma \mathcal{L}_N + (1-\gamma)\mathcal{L}_G$$

where γ: tunable hyperparameter

Task embedder　　Graph embedder



(b) Loss weight coefficient γ