

Continual Learning on Graphs: Challenges, Solutions, and Opportunities



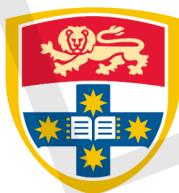
Dacheng TAO



Dongjin SONG



Xikun ZHANG



THE UNIVERSITY OF
SYDNEY

UCONN
UNIVERSITY OF CONNECTICUT

Agenda

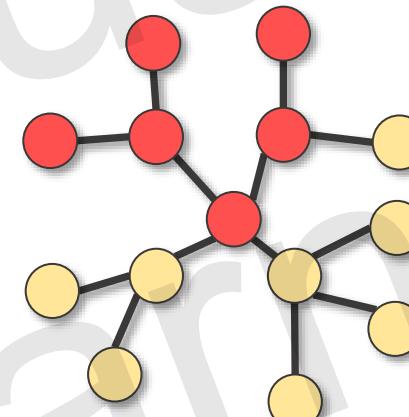
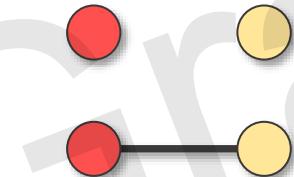
- Background on Graph Representation Learning (10 min)
- Motivation of Continual Graph Learning (10 min)
- Problem setup of CL and CGL (20 min)
- CL techniques & CGL techniques (60 min)
- Evaluation Metrics & CGL benchmarks (20 min)
- Future directions (10 min)
- Q&A

Agenda

- Background on Graph Representation Learning
- Motivation of Continual Graph Learning
- Problem setup of CL and CGL
- CL techniques & CGL techniques
- Evaluation Metrics & CGL benchmarks
- Future directions
- Q&A

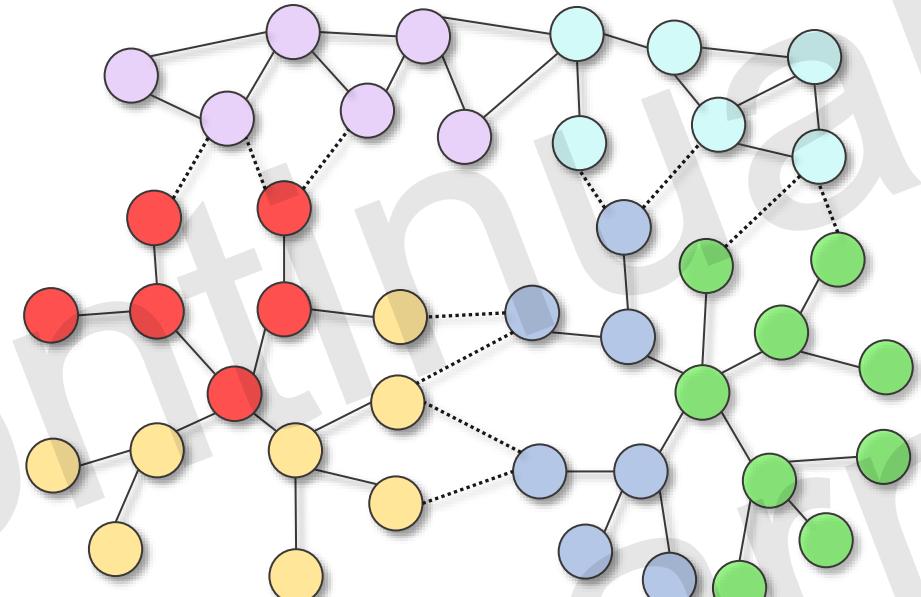
Graphs consist of

- Nodes (or vertices)
- Edges (or links) connecting nodes



Node representation learning via Graph Neural Networks (GNNs)

- class 1
- class 2
- class 3
- class 4
- class 5
- class 6



GNN

Representation
of each node

*Node integration
(optional)*



Representation of
the entire graph

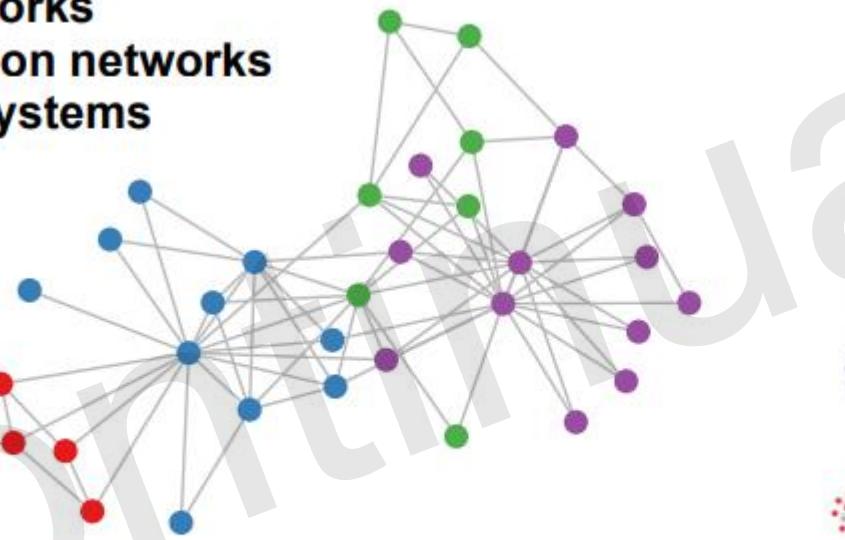
Graph data are pervasive in our world

Social networks

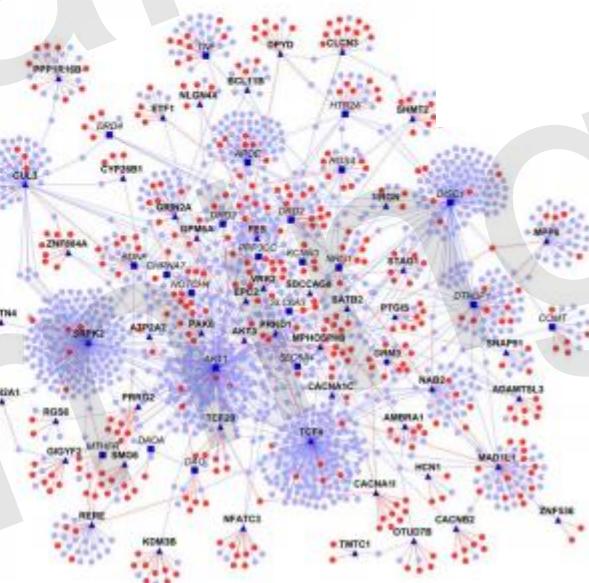
Citation networks

Communication networks

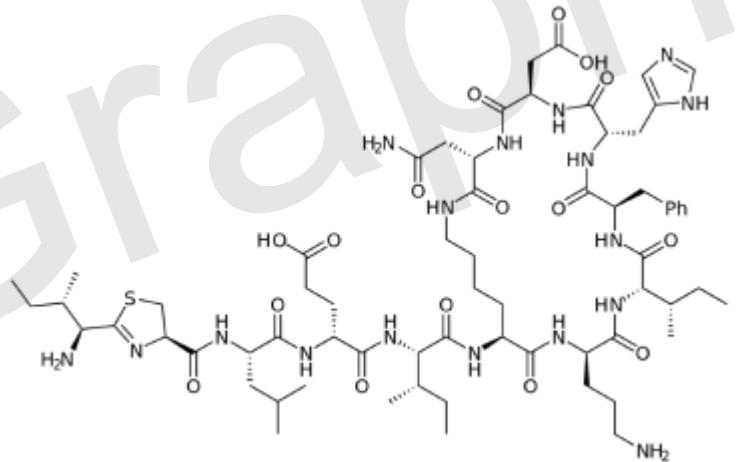
Multi-agent systems



Protein interaction networks



Molecules

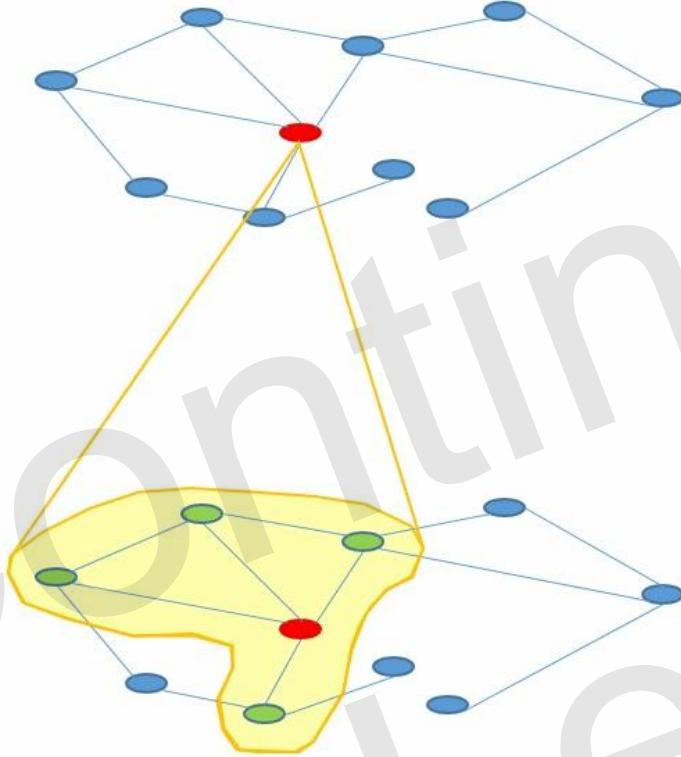


Left: <https://dilinikarunaratna.medium.com/introduction-to-graph-convolutional-networks-gcn-2235ed69875d>

Middle: <https://www.genengnews.com/insights/protein-protein-interactions-get-a-new-groove-on/>

Bright: <https://sl.wikipedia.org/wiki/Bacitracin>

Message Passing Neural Network (MPNN)



Continual Learning

Message function

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

Neighbours of v

Node update function

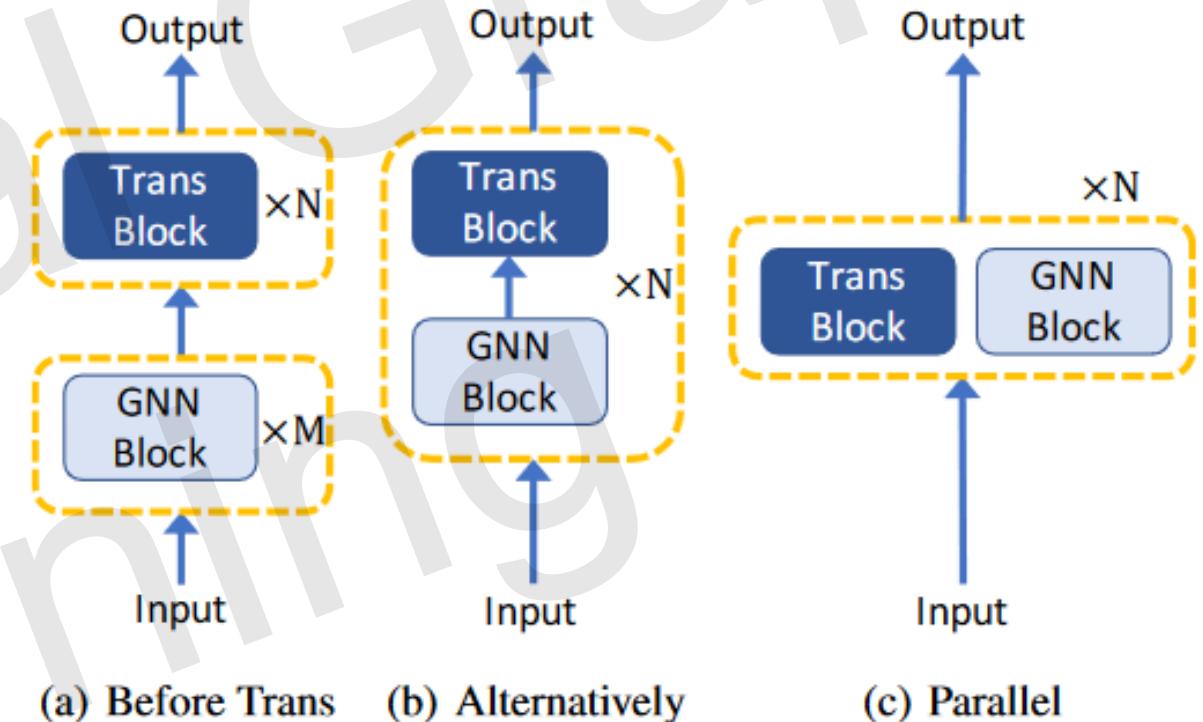
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

Readout function

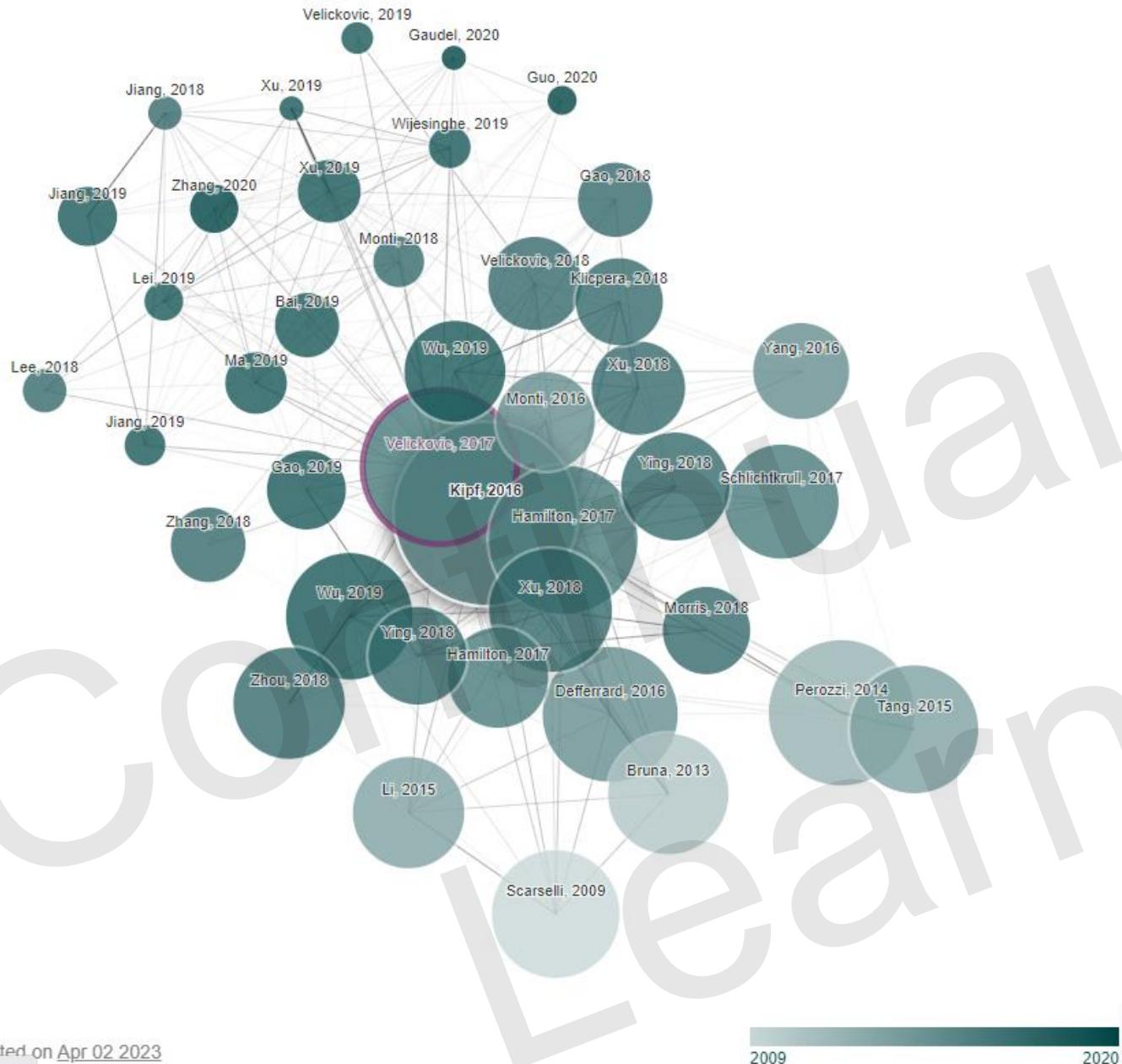
$$\hat{y} = R(\{h_v^T \mid v \in G\}).$$

Graph transformers

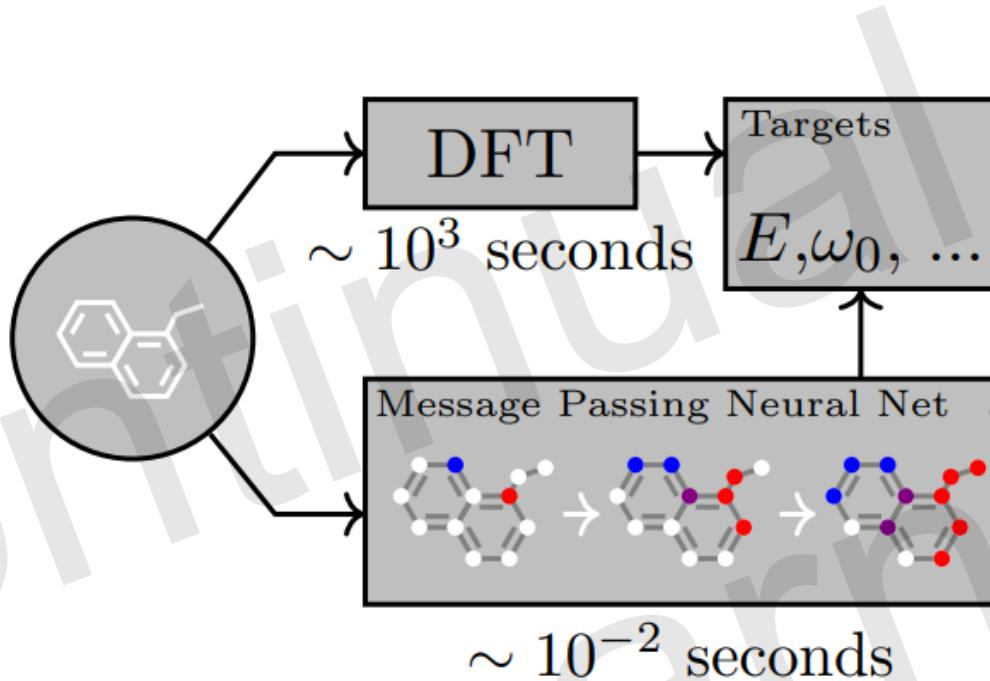
- GNNs as Auxiliary Modules in Transformer
- Improved Positional Embeddings from Graphs
- Improved Attention Matrices from Graphs



Popular node-level task:
Node classification on
Citation network

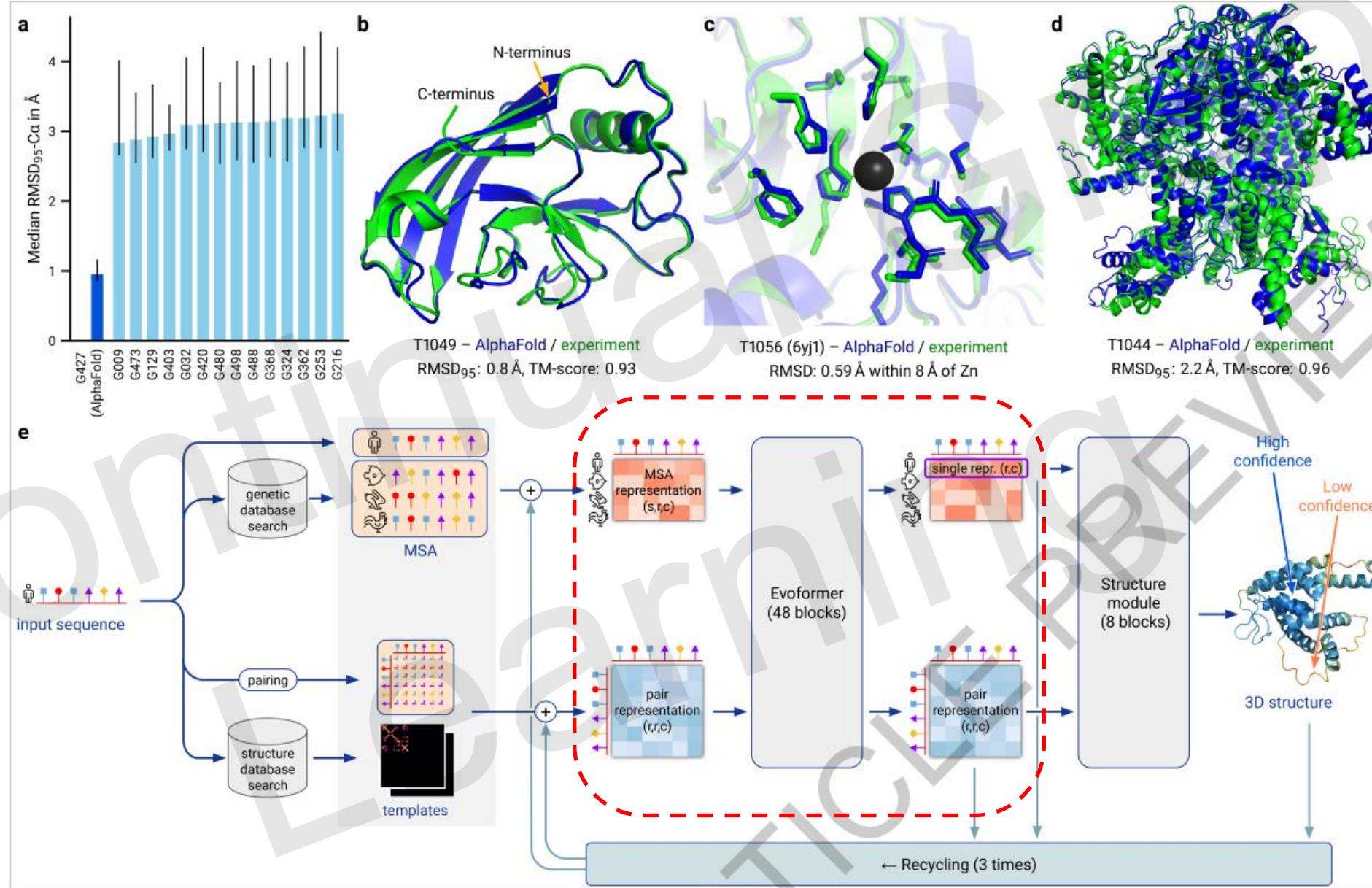


Message Passing Neural Network (MPNN)

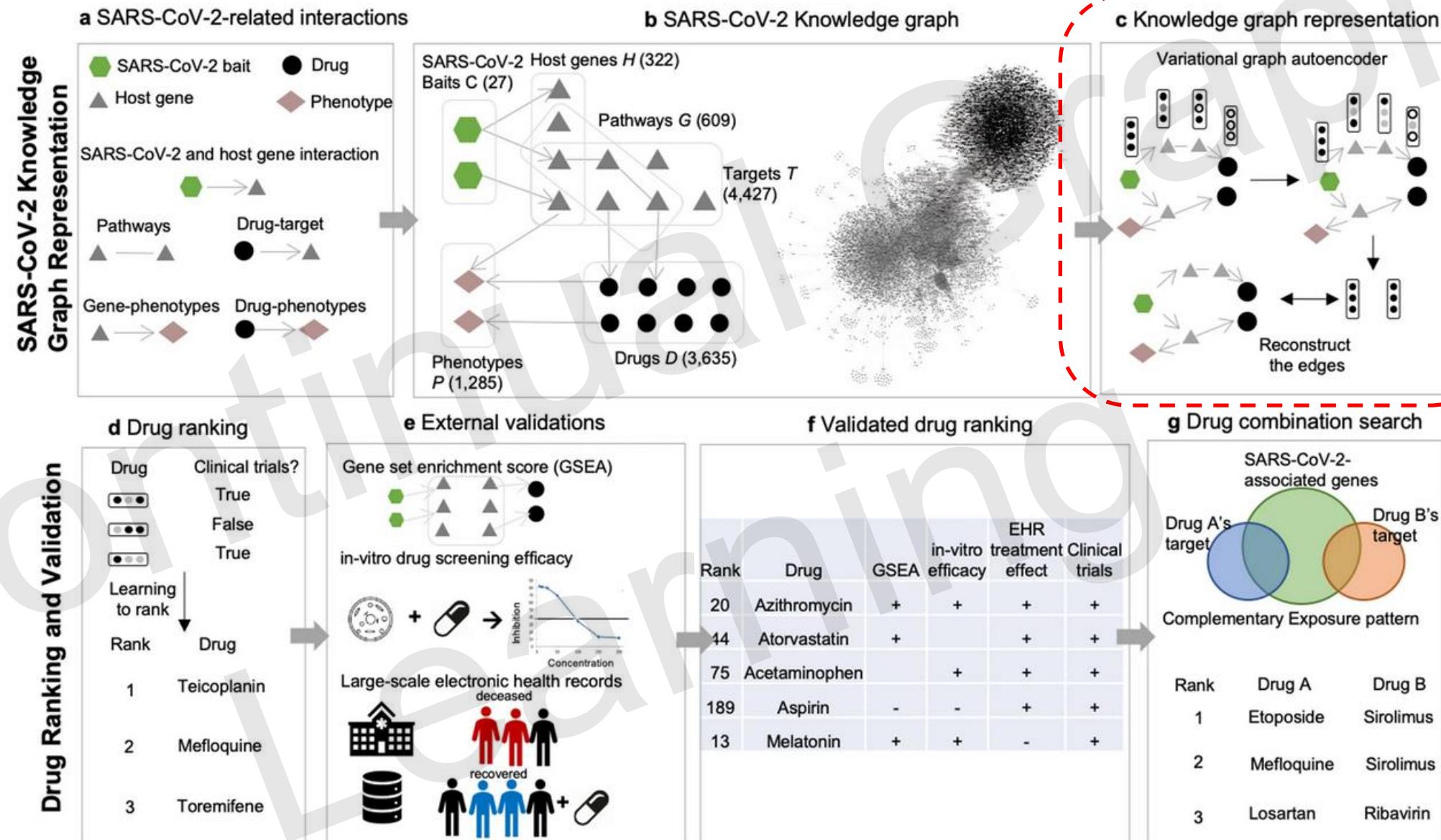


Popular graph-level
task: Molecule
property prediction

AlphaFold2 (Protein structure prediction)

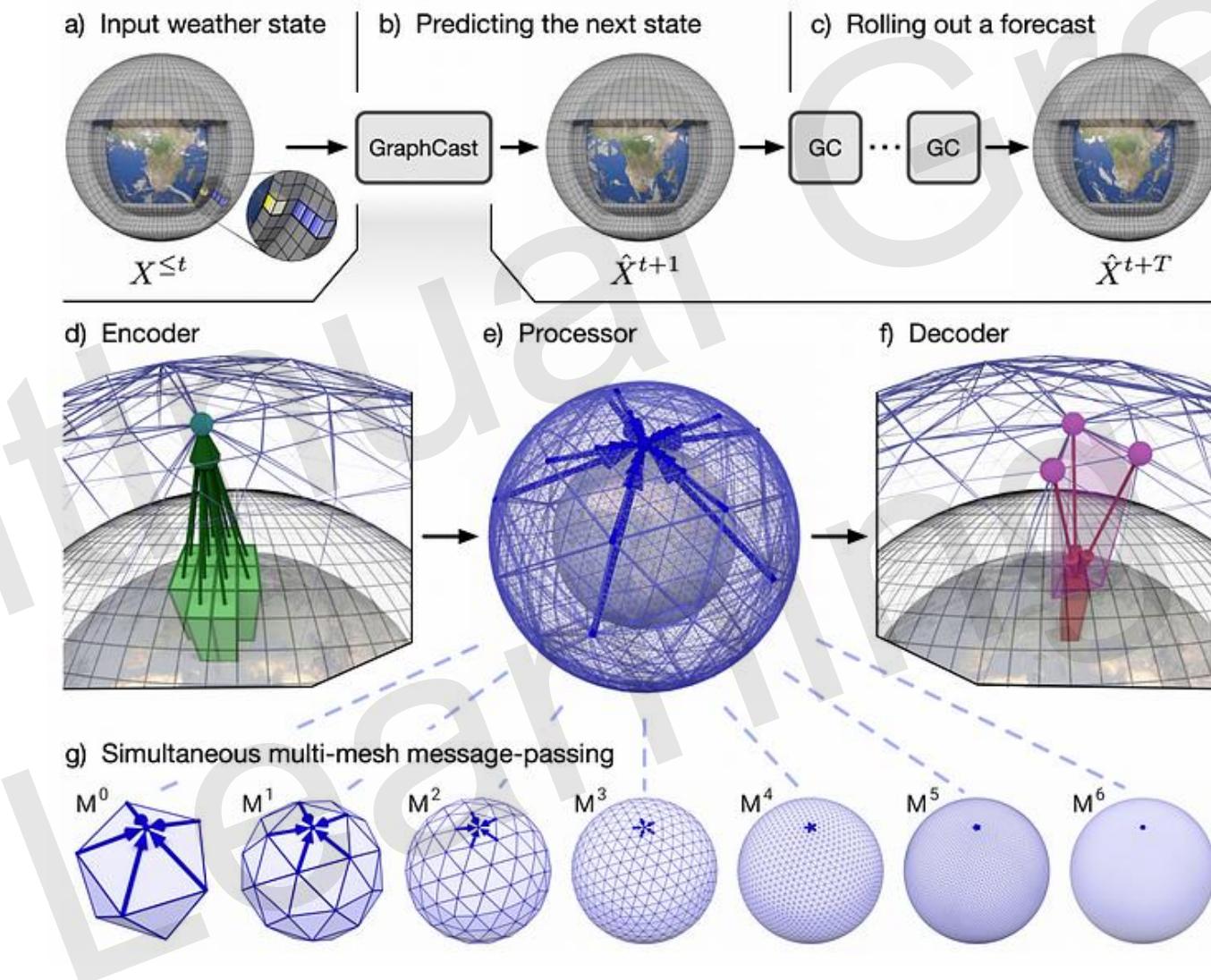


Drug discovery



Hsieh, Kanglin, et al. "Drug repurposing for COVID-19 using graph neural network and harmonizing multiple evidence." Scientific reports 11.1 (2021): 23179.

GraphCast (Weather forecasting)

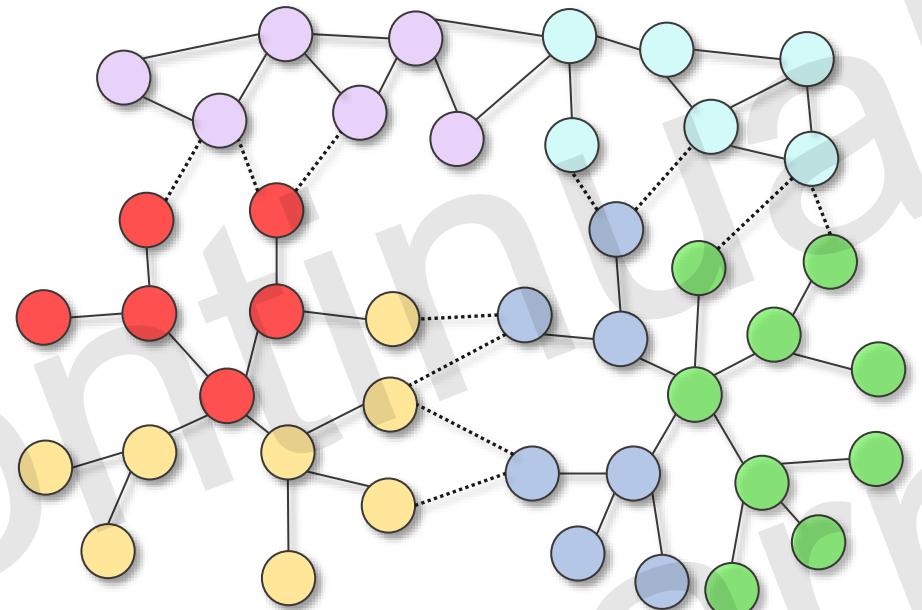


Agenda

- Background on Graph Representation Learning
- Motivation of Continual Graph Learning
- Problem setup of CL and CGL
- CL techniques & CGL techniques
- Evaluation Metrics & CGL benchmarks
- Future directions

Learning on static graphs

- class 1
- class 2
- class 3
- class 4
- class 5
- class 6

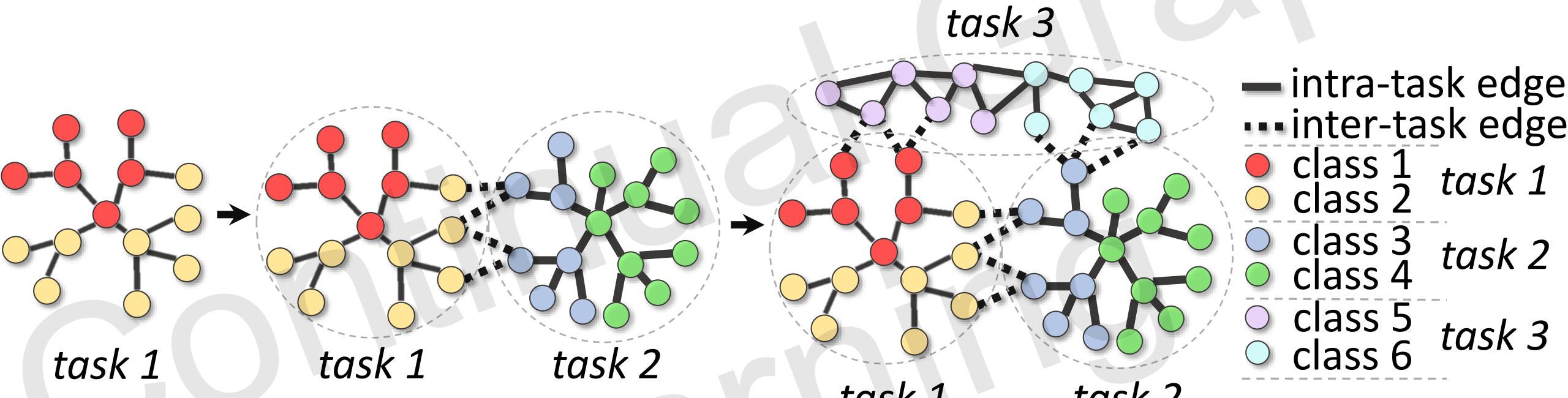


Model

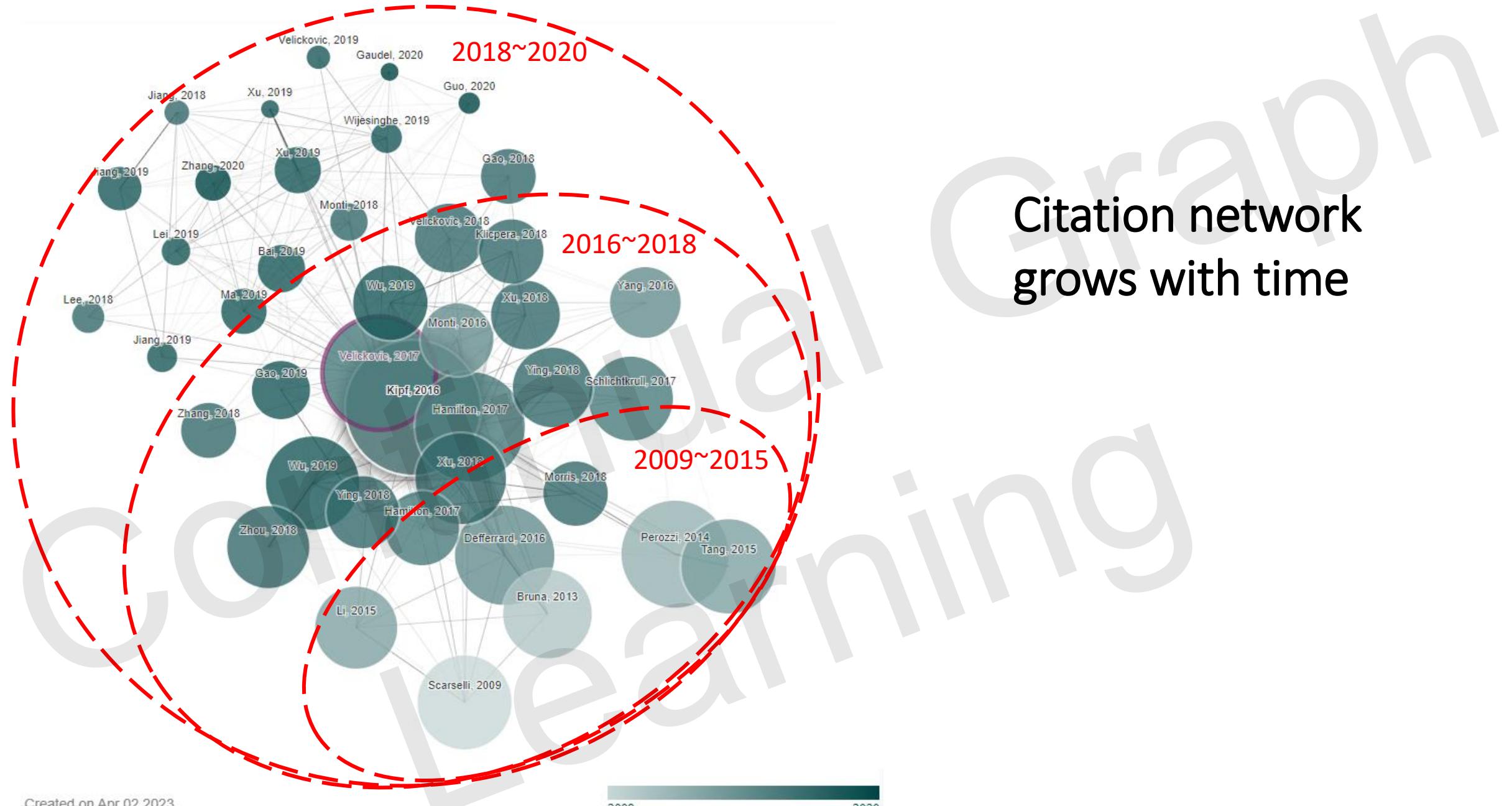


Representation
of each node

Real world graphs are expanding all the time

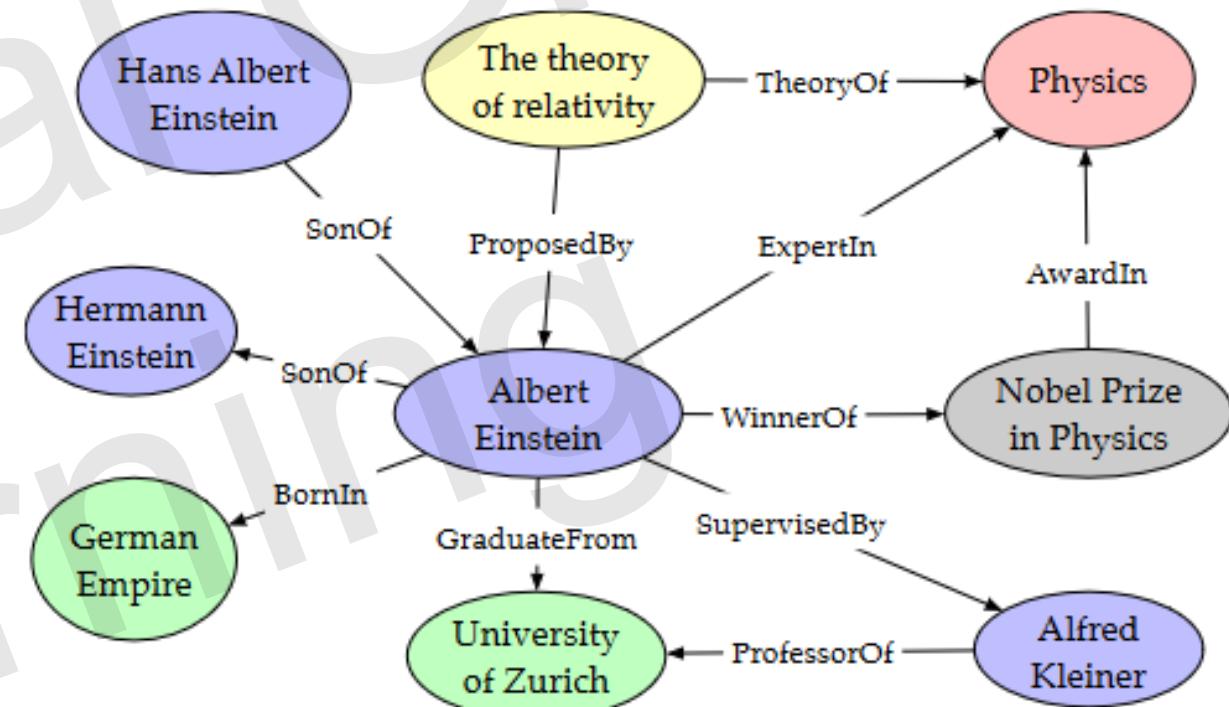


Citation network
grows with time

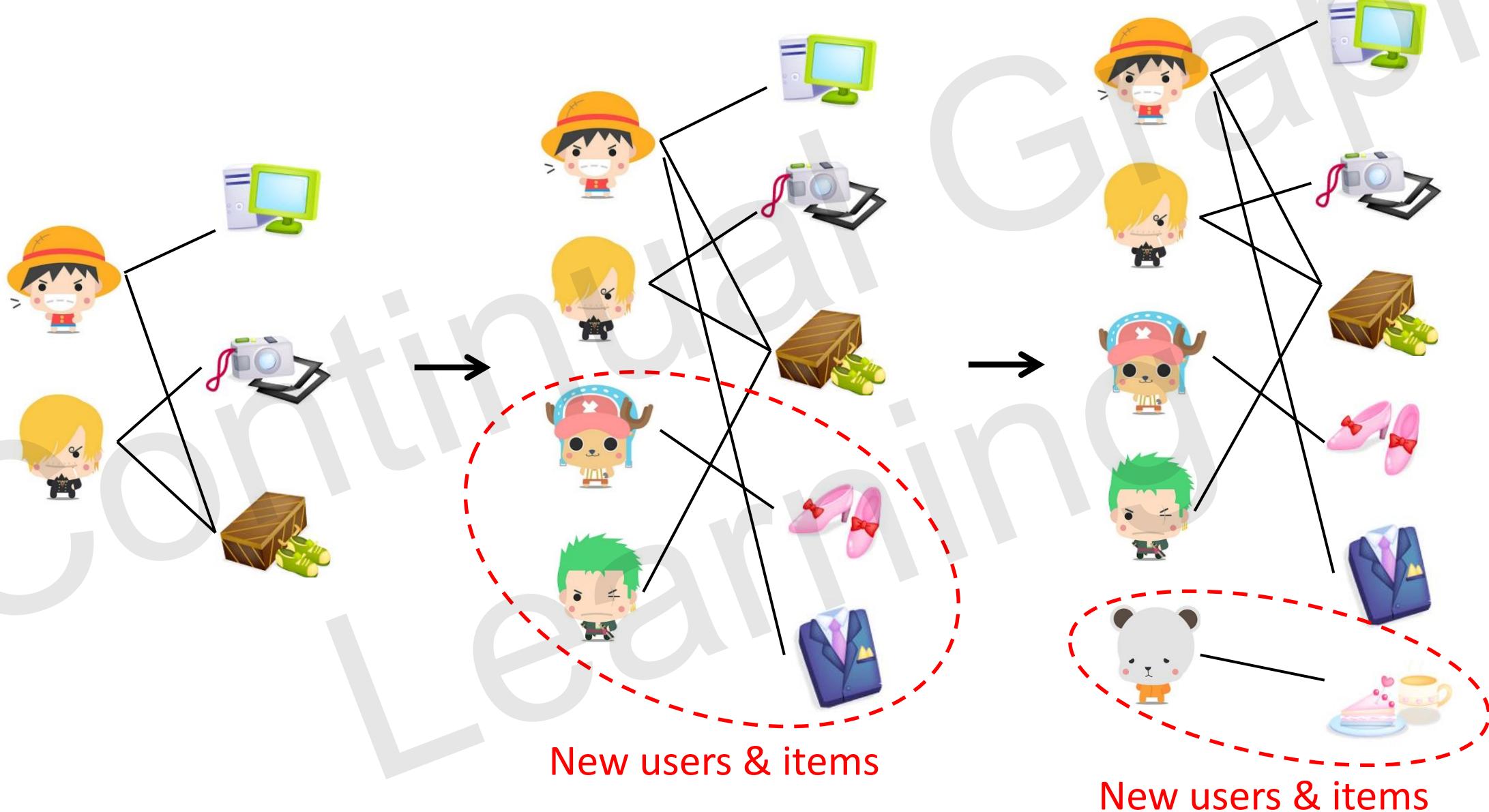


Knowledge graph grows with time

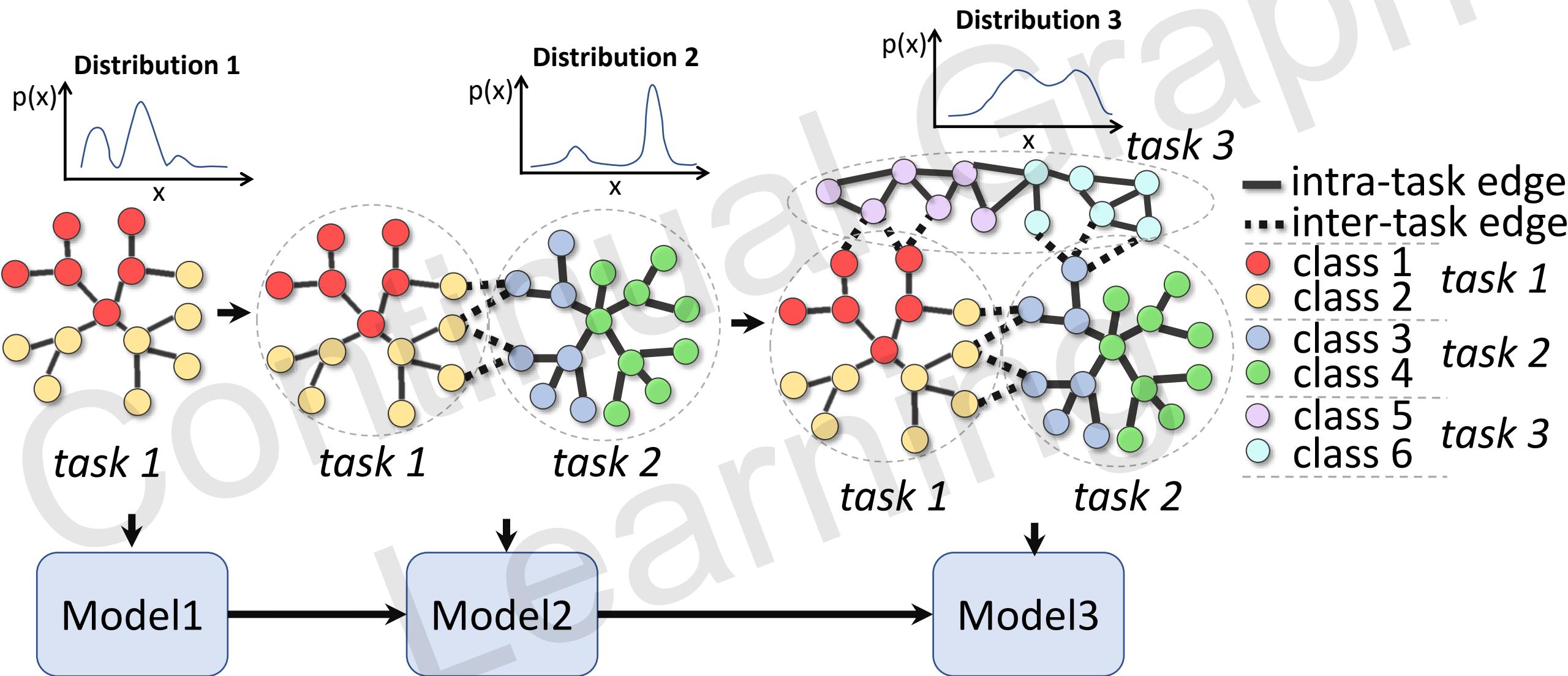
(Albert Einstein, **BornIn**, German Empire)
(Albert Einstein, **SonOf**, Hermann Einstein)
(Albert Einstein, **GraduateFrom**, University of Zurich)
(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)
 (Albert Einstein, **ExpertIn**, Physics)
(Nobel Prize in Physics, **AwardIn**, Physics)
(The theory of relativity, **TheoryOf**, Physics)
(Albert Einstein, **SupervisedBy**, Alfred Kleiner)
(Alfred Kleiner, **ProfessorOf**, University of Zurich)
(The theory of relativity, **ProposedBy**, Albert Einstein)
 (Hans Albert Einstein, **SonOf**, Albert Einstein)



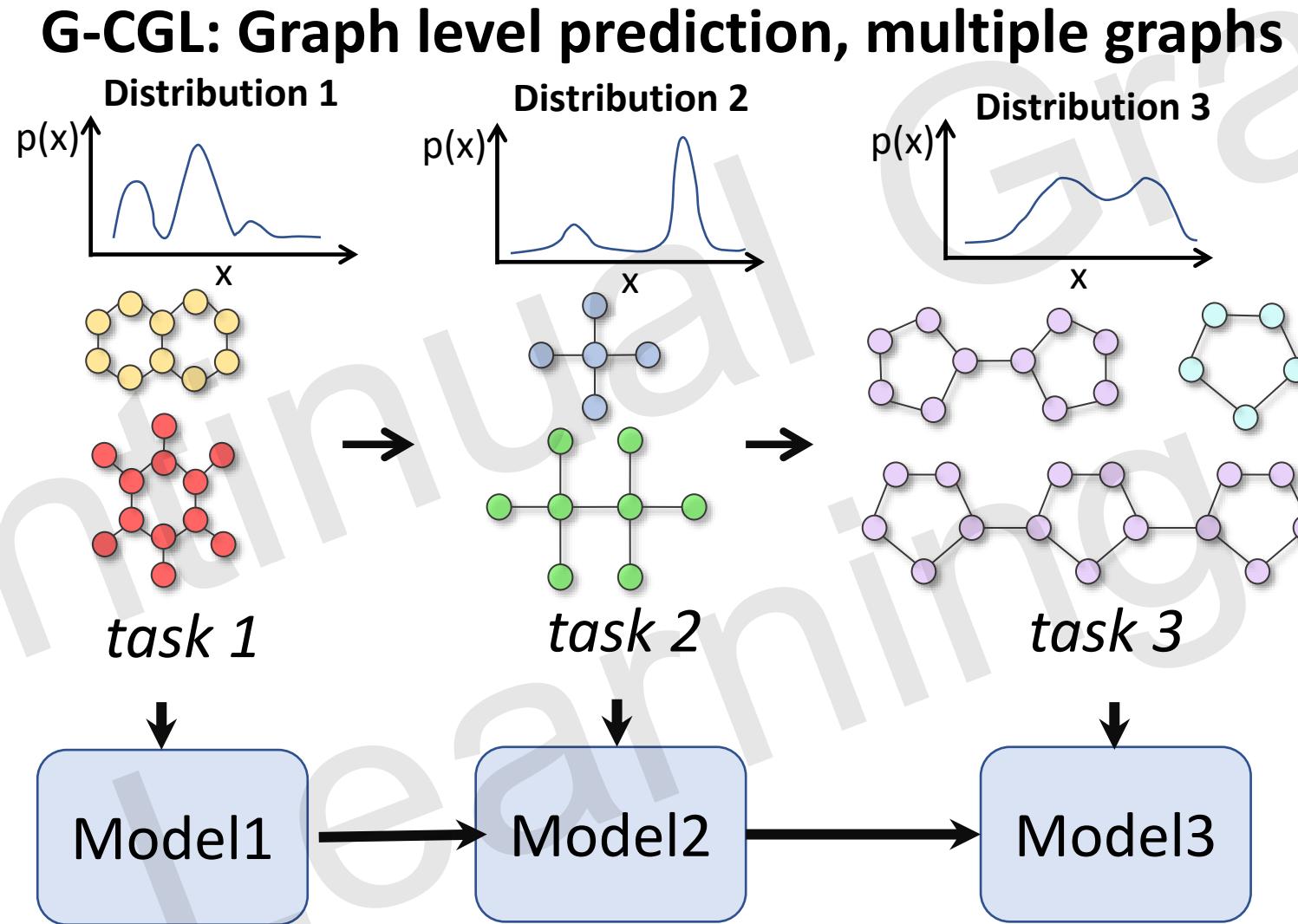
Recommender system graph grows with time



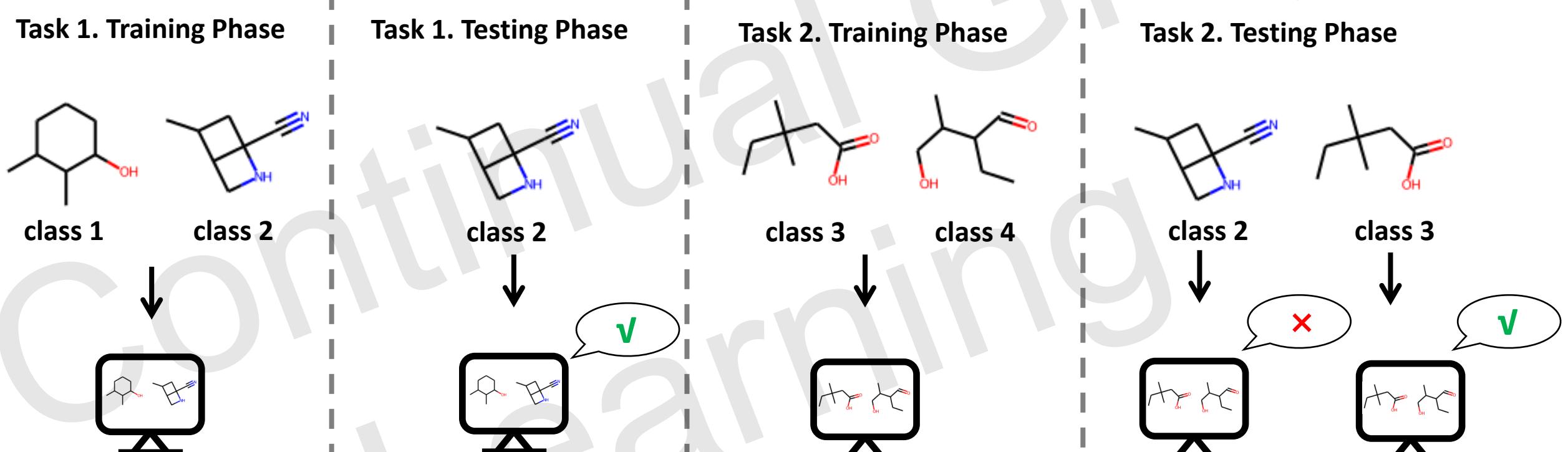
Model has to adapt



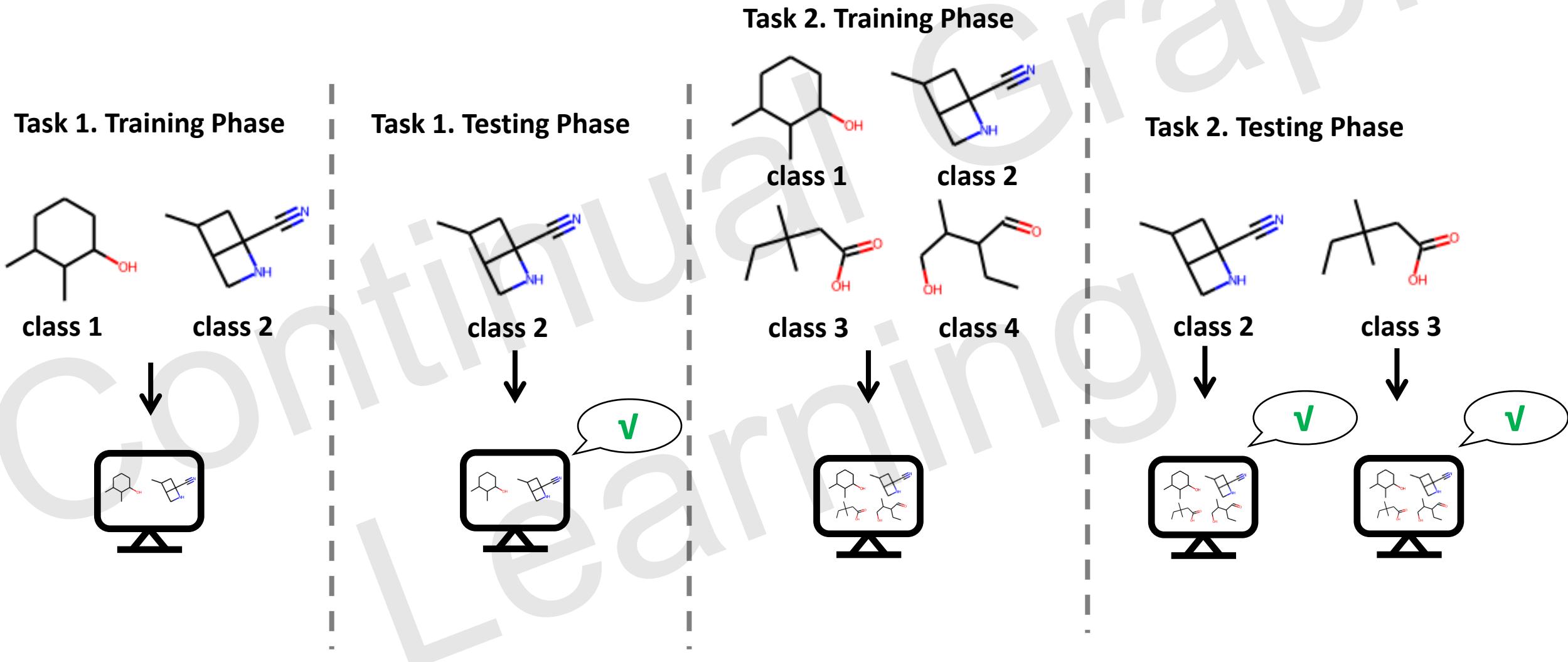
Realistic continual learning scenario



Catastrophic forgetting



Can we just retrain over all data?



Retraining is computationally impractical

Scale	Name	#Nodes	#Edges*
Medium	ogbn-products	2,449,029	61,859,140
Medium	ogbn-proteins	132,534	39,561,252
Small	ogbn-arxiv	169,343	1,166,243
Large	ogbn-papers100M	111,059,956	1,615,685,872
Medium	ogbn-mag	1,939,743	21,111,007

Scale	Name	#Graphs	#Nodes per graph	#Edges per graph*
Small	ogbg-molhiv	41,127	25.5	27.5
Medium	ogbg-molpcba	437,929	26.0	28.1
Medium	ogbg-ppa	158,100	243.4	2,266.1
Medium	ogbg-code2	452,741	125.2	124.2

Scale	Name	#Nodes	#Edges*
Medium	ogbl-ppa	576,289	30,326,273
Small	ogbl-collab	235,868	1,285,465
Small	ogbl-ddi	4,267	1,334,889
Medium	ogbl-citation2	2,927,963	30,561,187
Medium	ogbl-wikig2	2,500,604	17,137,181
Small	ogbl-biokg	93,773	5,088,434
Medium	ogbl-vessel*	3,538,495	5,345,897

Task category	Name	#Graphs	#Total nodes	#Total edges
Node-level	MAG240M	1	244,160,499	1,728,364,232
Link-level	WikiKG90Mv2	1	91,230,610	601,062,811
Graph-level	PCQM4Mv2	3,746,619	52,970,652	54,546,813

Objectives

- Enough plasticity to adapt to new tasks (including new topologies)
- Minimal forgetting on previous tasks (topologies)
- If possible, positive cross task transfer

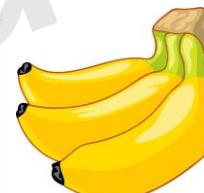
Agenda

- Background on Graph Representation Learning
- Motivation of Continual Graph Learning
- Problem setup of CL and CGL
- CL techniques & CGL techniques
- Evaluation Metrics & CGL benchmarks
- Future directions

Basic concept: tasks



Task 1

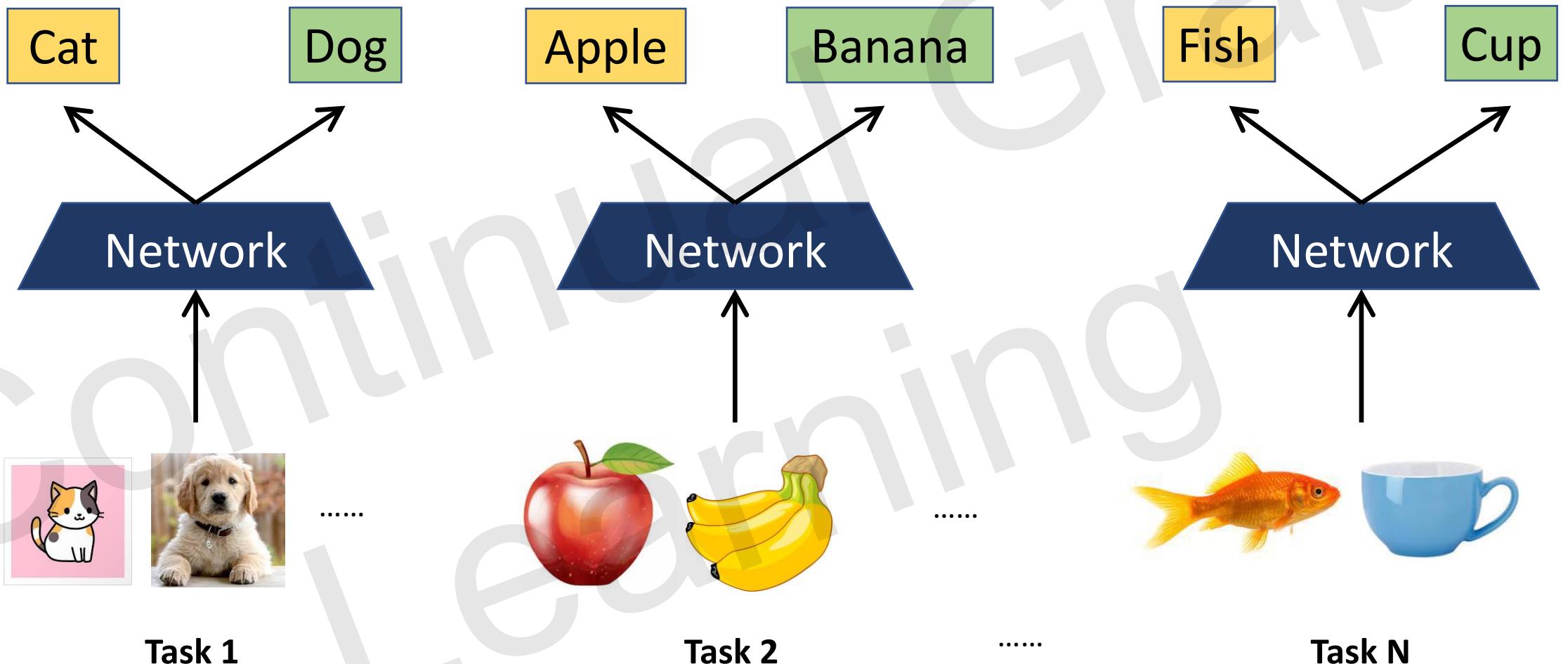


Task 2

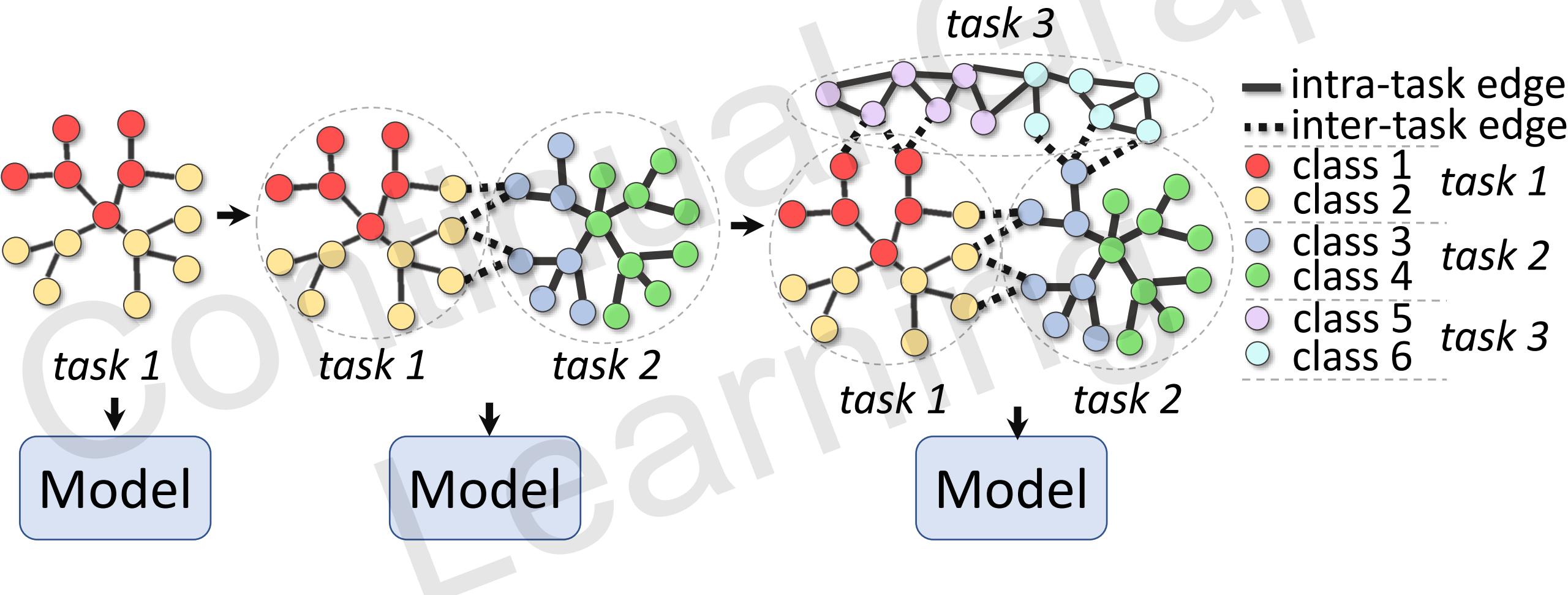


Task N

Commonly adopted task constructions

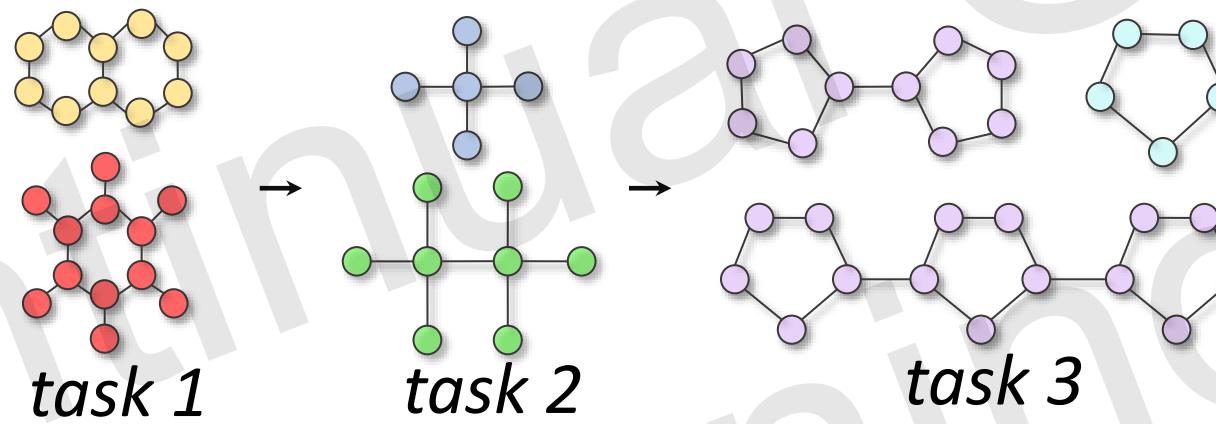


Realistic continual learning scenario



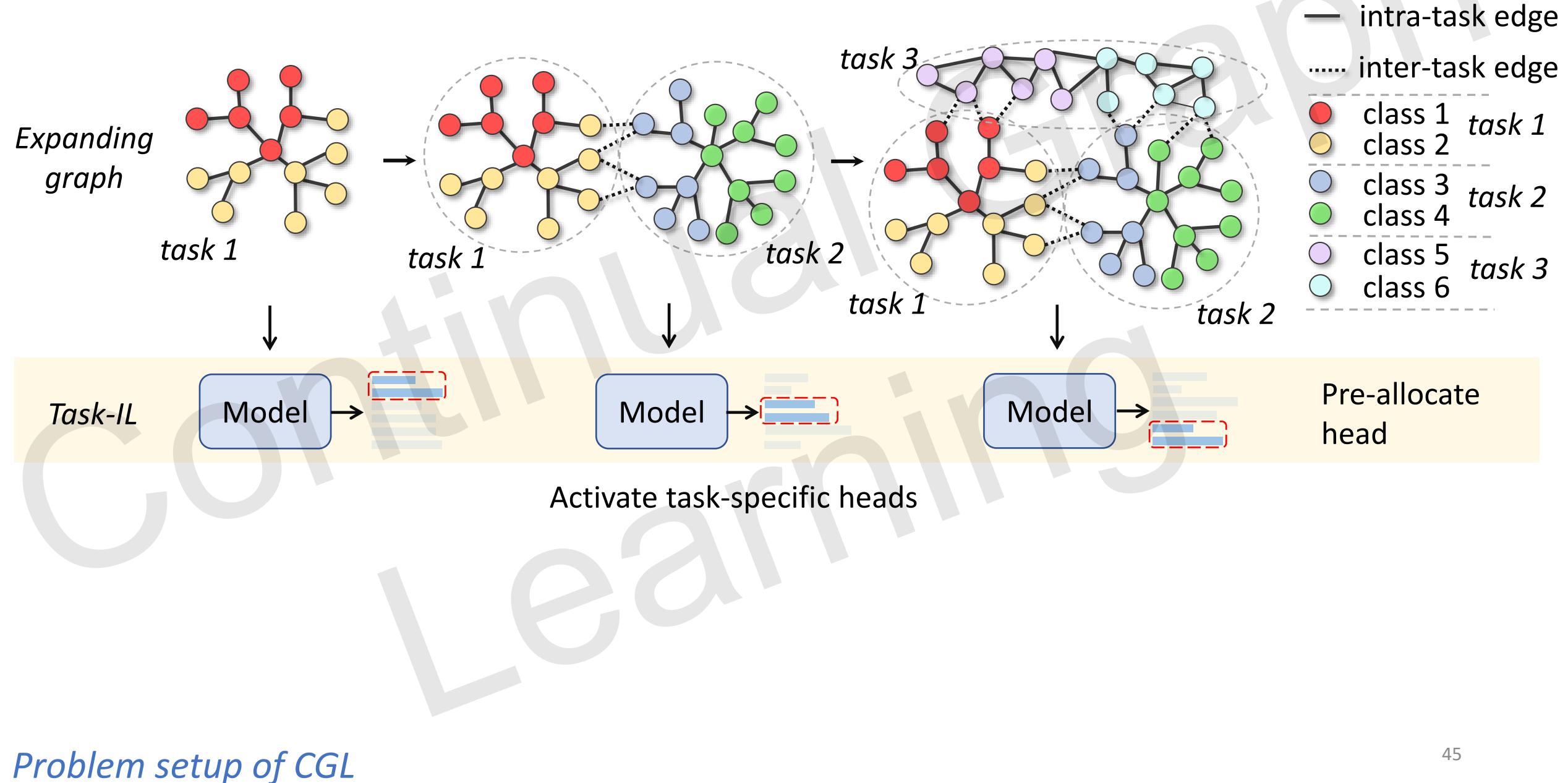
Realistic continual learning scenario

G-CGL: Graph level prediction, multiple graphs

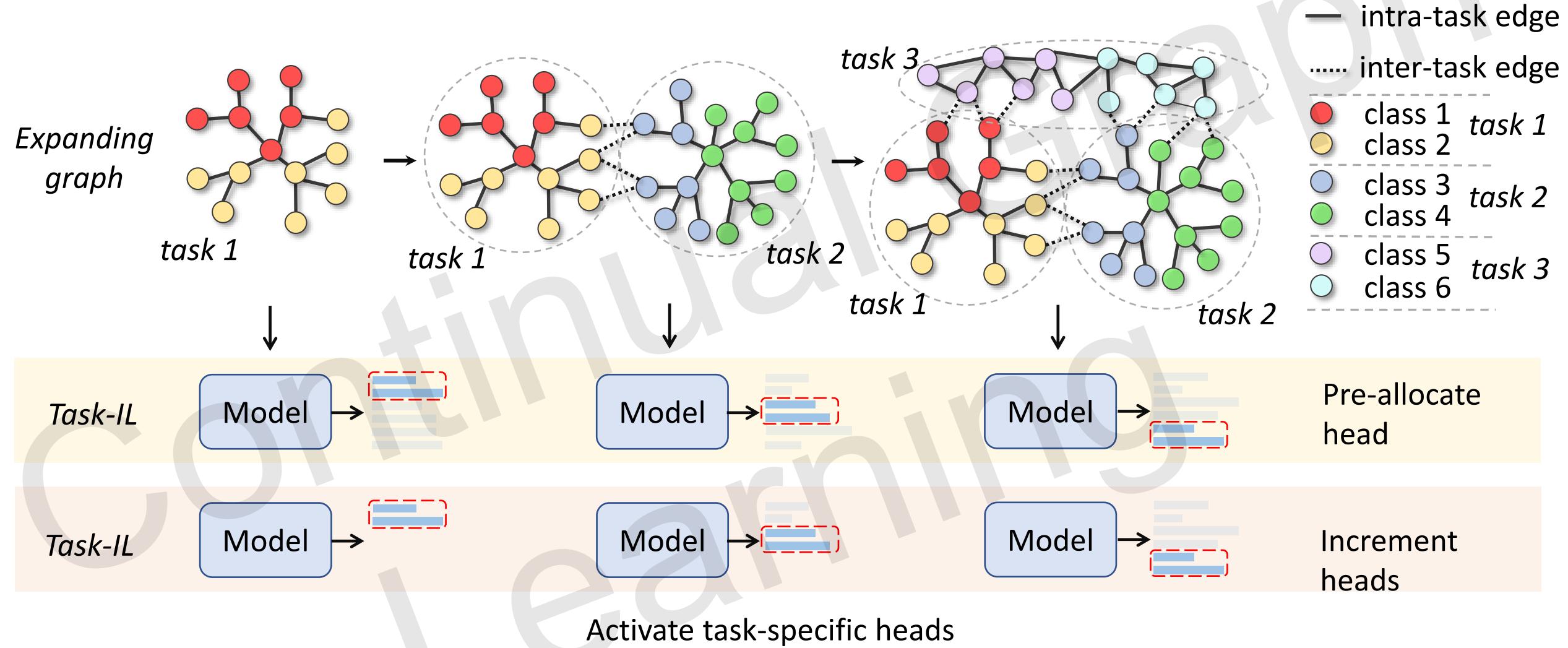


- Node- or graph-level tasks?

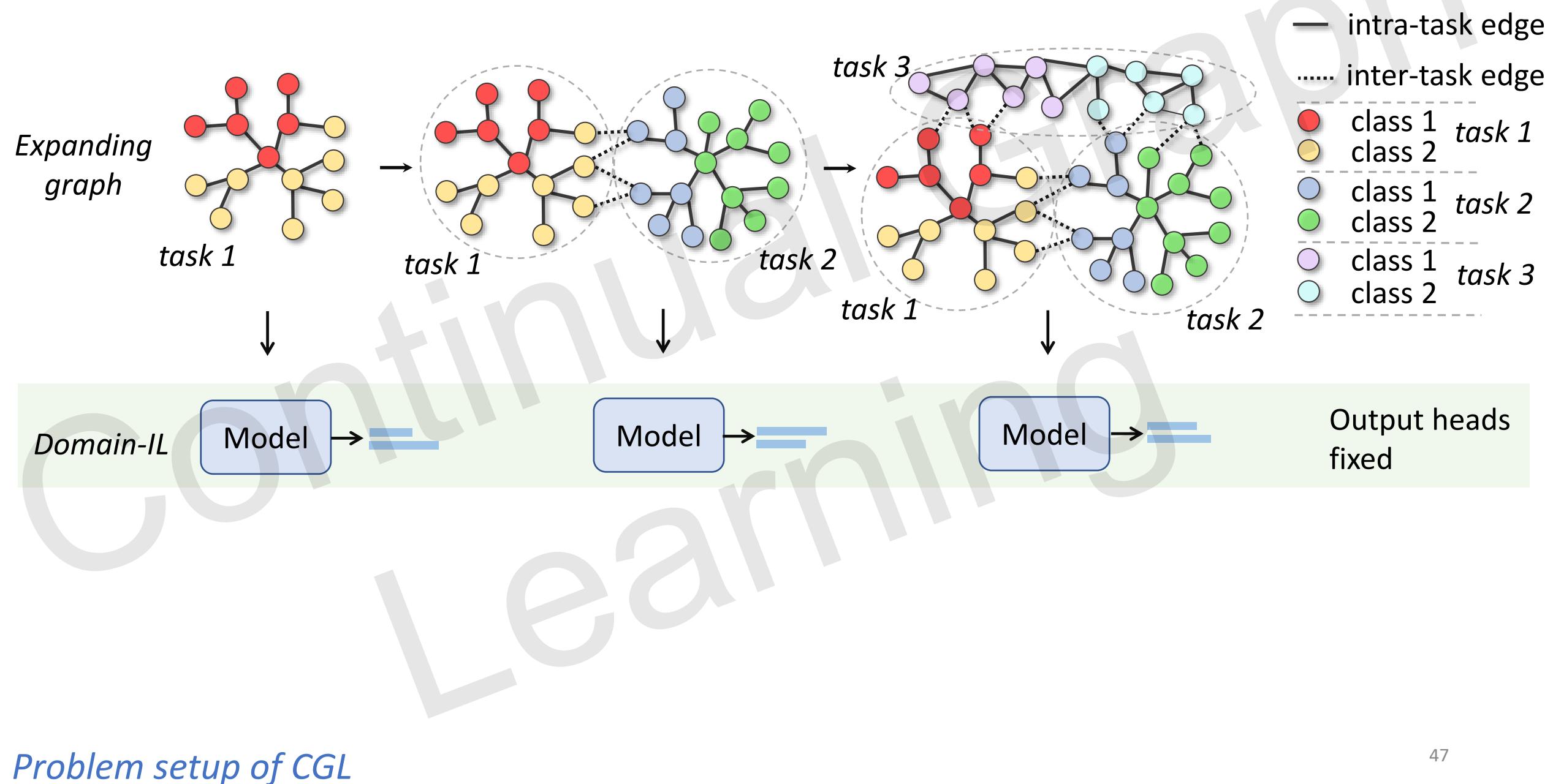
Task-IL & Node-level tasks



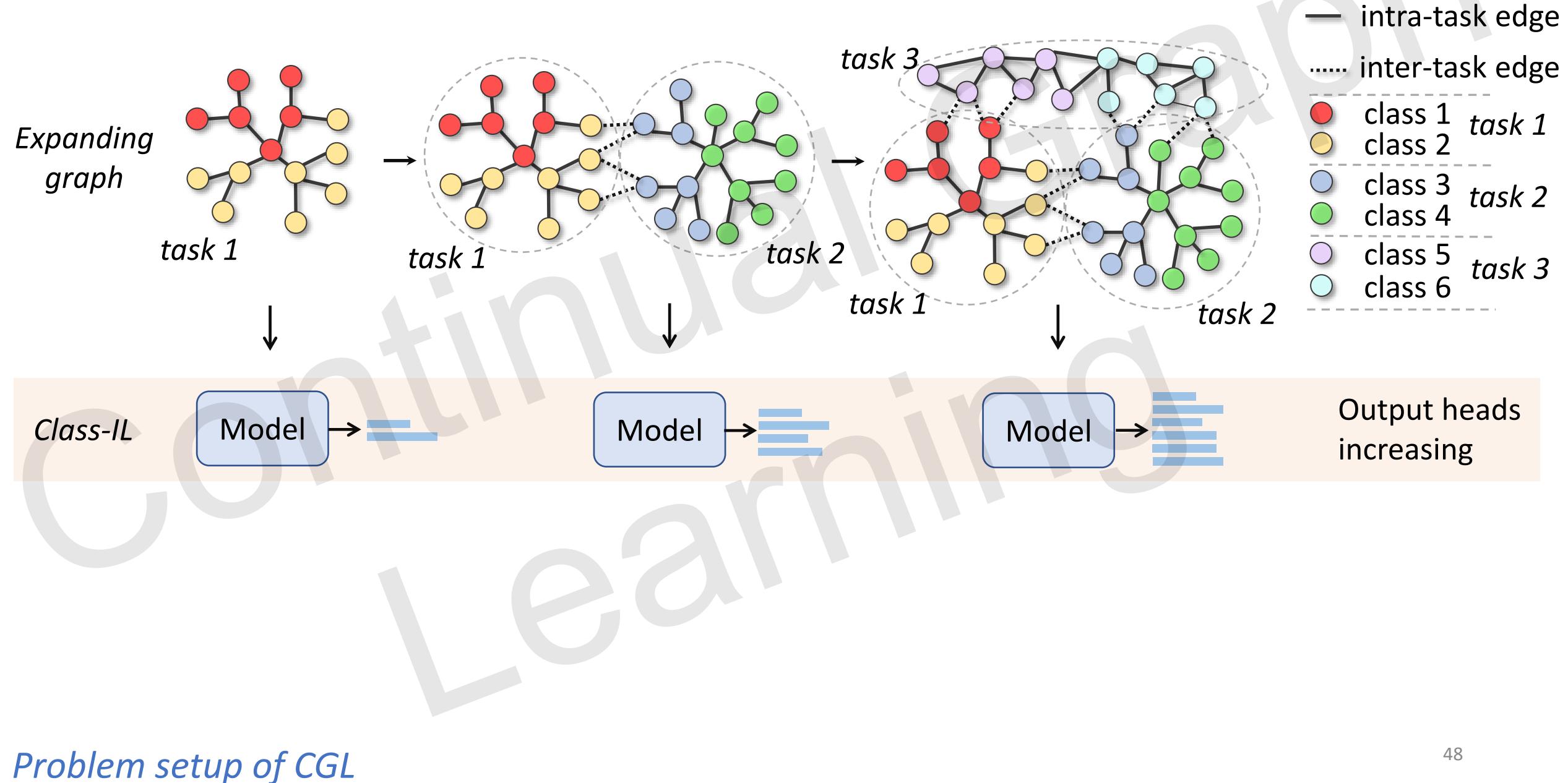
Task-IL & Node-level tasks



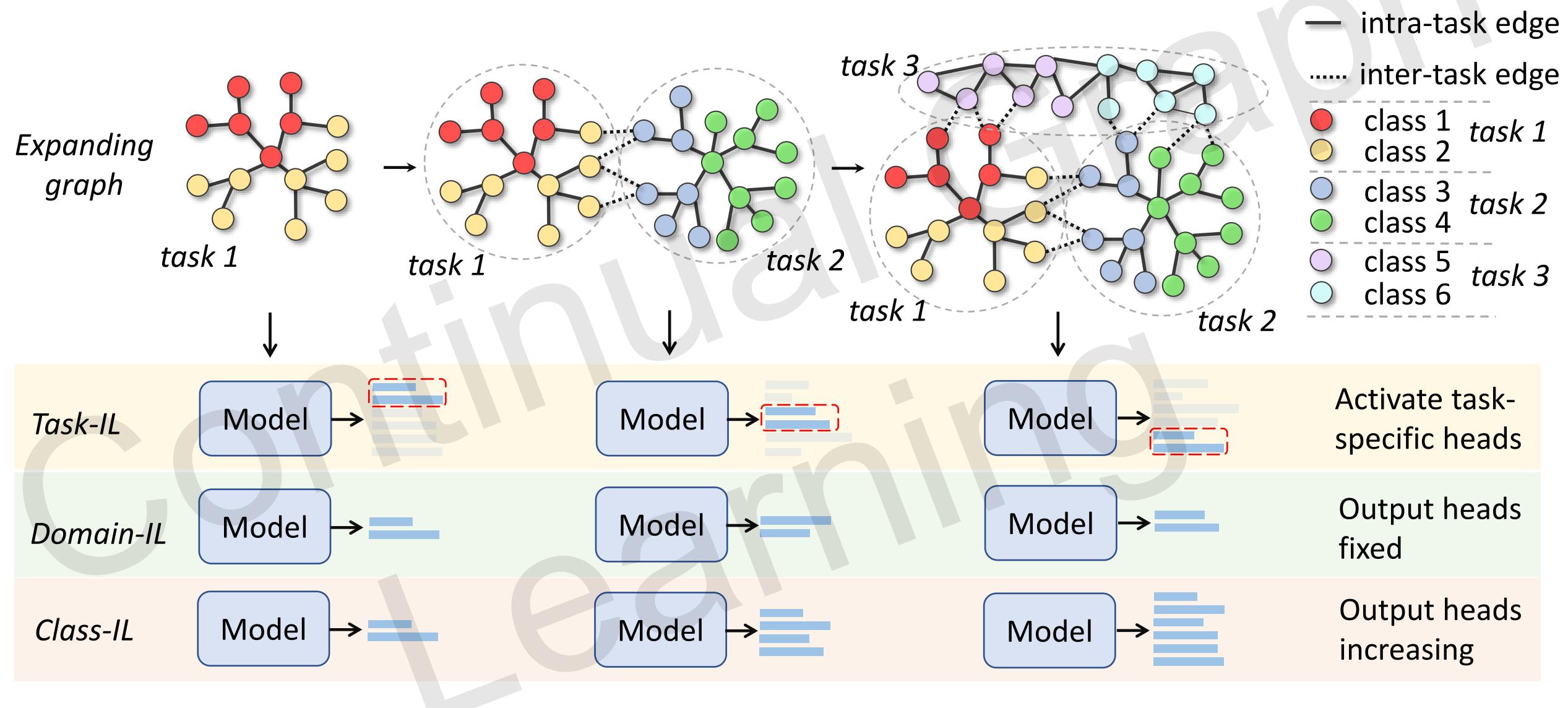
Domain-IL & Node-level tasks



Class-IL & Node-level tasks



Summary of three incremental scenarios



Difficulty: Plasticity-stability dilemma

- Plasticity: adapting to new tasks
- Stability: maintaining performance on old tasks
- Catastrophic forgetting: models adapting well to new tasks while forgetting old tasks severely

Agenda

- Background on Graph Representation Learning
- Motivation of Continual Graph Learning
- Problem setup of CL and CGL
- CL techniques & CGL techniques
- Evaluation Metrics & CGL benchmarks
- Future directions

Rough Categories

- **Regularization:** Penalize changes to the model via regularizations
- **Parameter-isolation:** Separate parameters for new and old tasks (partially or entirely)
- **Memory-replay:** Replay data from existing/old tasks while adapting the model to new ones

Rough Categories

- **Regularization:** Penalize changes to the model via regularizations
- **Parameter-isolation:** Separate parameters for new and old tasks (partially or entirely)
- **Memory-replay:** Replay old task data to the model when learning new ones

Elastic weight consolidation (EWC, regularization based)

Basic loss for learning
the currently given task

$$\mathcal{L}(\theta) = \boxed{\mathcal{L}_B(\theta)} + \boxed{\sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{A,i}^*)^2}$$

Regularization term for
minimizing the
forgetting problem

λ :balances the contribution of the old
tasks

F_i :Parameter importance to task A
(measured by Fisher information)

Elastic weight consolidation (EWC, regularization based)

Basic loss for learning
the currently given task

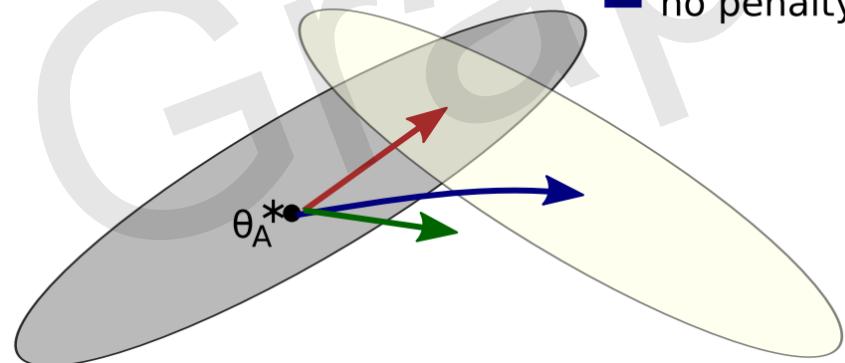
$$\mathcal{L}(\theta) = \boxed{\mathcal{L}_B(\theta)} + \boxed{\sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{A,i}^*)^2}$$

Regularization term for
minimizing the
forgetting problem

λ :balances the contribution of the old
tasks

F_i :Parameter importance to task A
(measured by Fisher information)

- Low error for task B
- Low error for task A



Elastic weight consolidation (EWC, regularization based)

Basic loss for learning
the currently given task

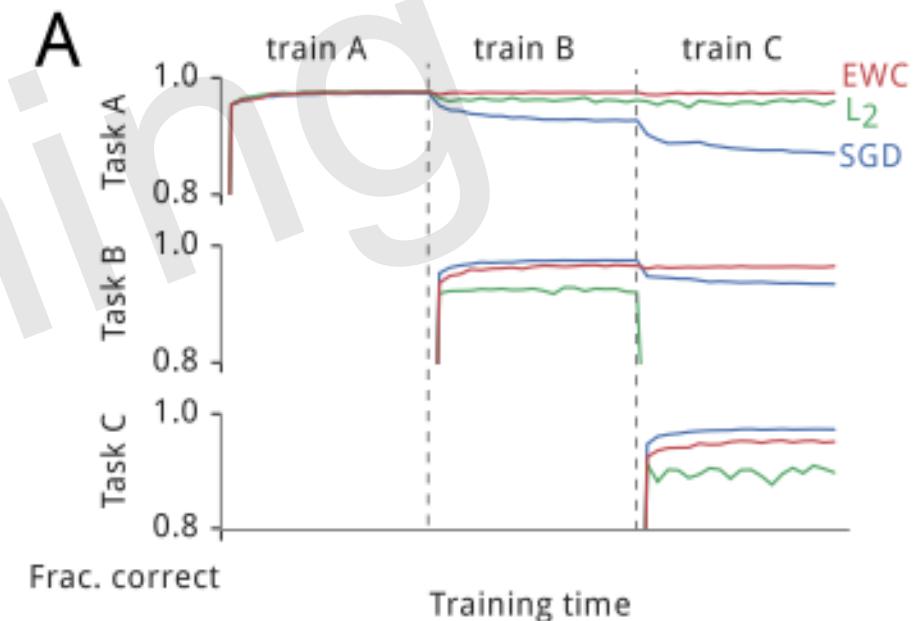
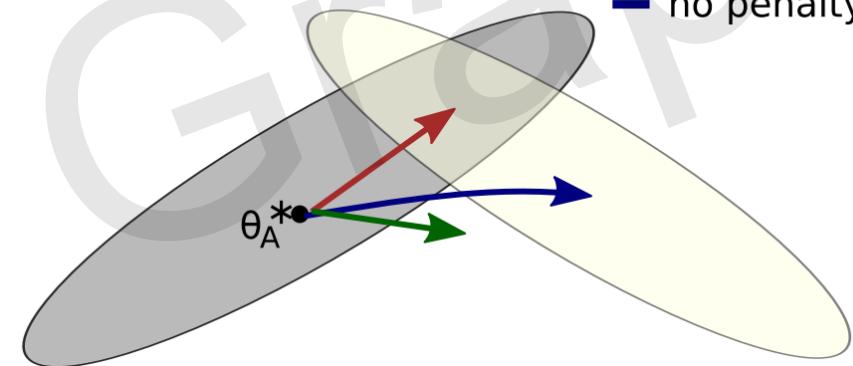
$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i (\theta_i - \theta_{A,i}^*)^2$$

Regularization term for
minimizing the
forgetting problem

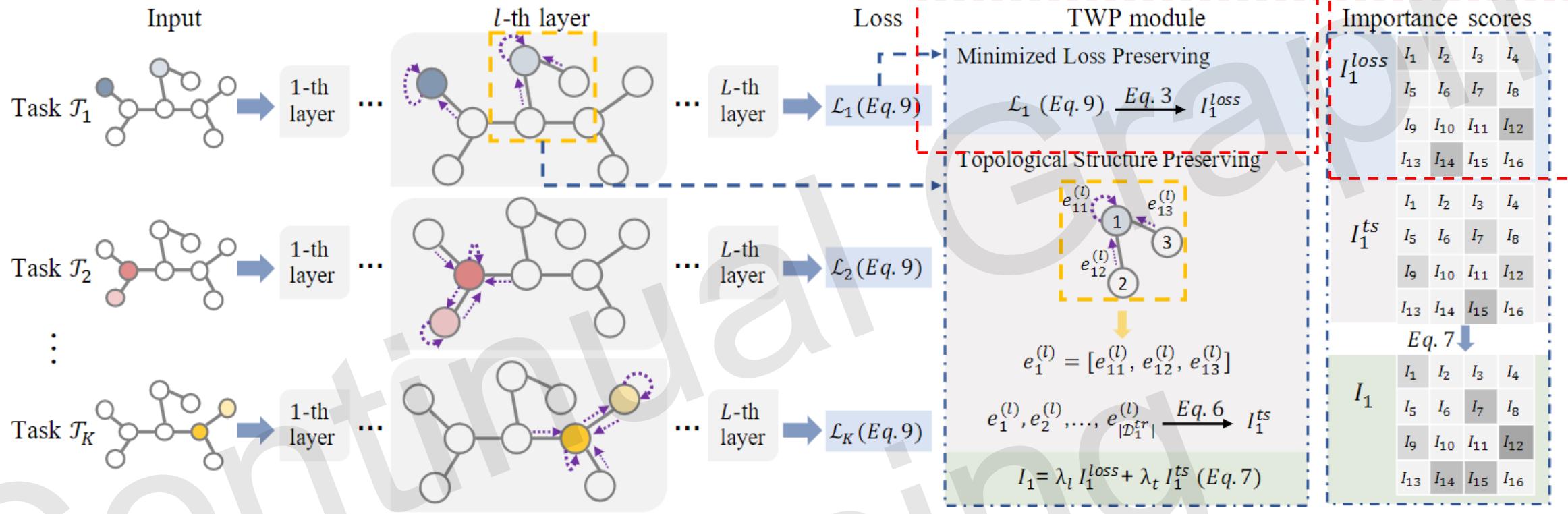
λ :balances the contribution of the old
tasks

F_i :Parameter importance to task A
(measured by Fisher information)

- Low error for task B
- Low error for task A



Topology-aware Weight Preserving (TWP, regularization based)



Minimized Loss Preserving

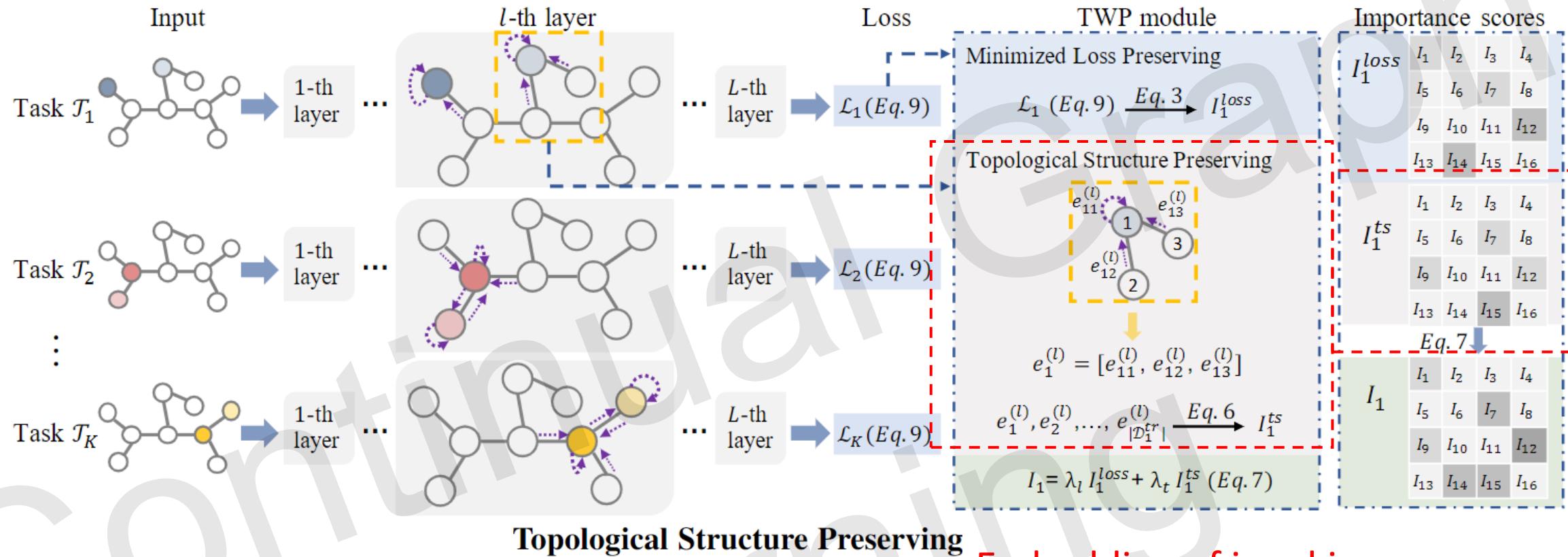
$$\mathcal{L}(X_k^{tr}; W + \Delta W) - \mathcal{L}(X_k^{tr}; W) \approx \sum_m f_m(X_k^{tr}) \Delta w_m$$

$$f_m(X_k^{tr}) = \frac{\partial \mathcal{L}}{\partial \mathbf{w}_m}$$

$$I_k^{loss} = [\|f_m(X_k^{tr})\|]$$

importance score (loss)

Topology-aware Weight Preserving (TWP, regularization based)



Topological Structure Preserving

$$e_{ij}^{(l)} = a(\mathbf{H}_{i,j}^{(l-1)}; W^{(l)})$$

(for GAT)

$$g_m(\mathbf{H}^{(l-1)}) = \frac{\partial \left(\left\| [e_1^{(l)}, \dots, e_{|\mathcal{D}_k^{tr}|}^{(l)}] \right\|_2^2 \right)}{\partial w_m}$$

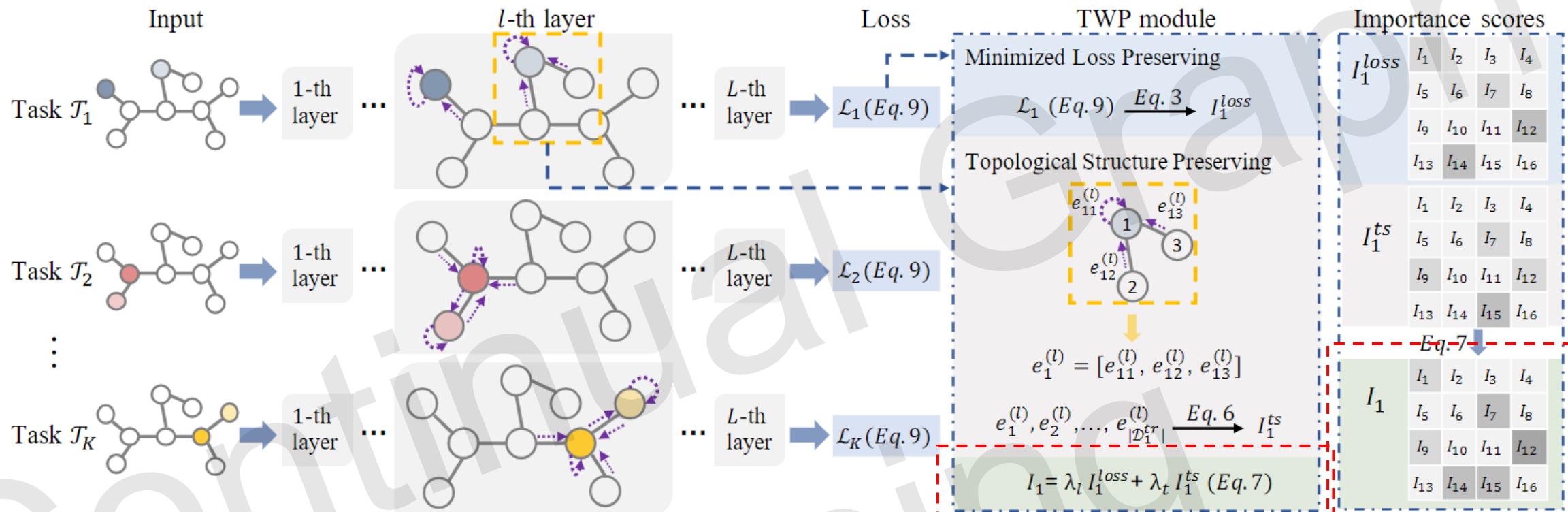
$$e_{ij}^{(l)} = (\mathbf{h}_i^{(l-1)} W^{(l)})^T \tanh(\mathbf{h}_j^{(l-1)} W^{(l)})$$

(for any GNN)

$$I_k^{ts} = [\|g_m(\mathbf{H}_k^{(l-1)})\|]$$

importance score
(topology)

Topology-aware Weight Preserving (TWP, regularization based)



$$I_k = \lambda_l I_k^{loss} + \lambda_t I_k^{ts}$$

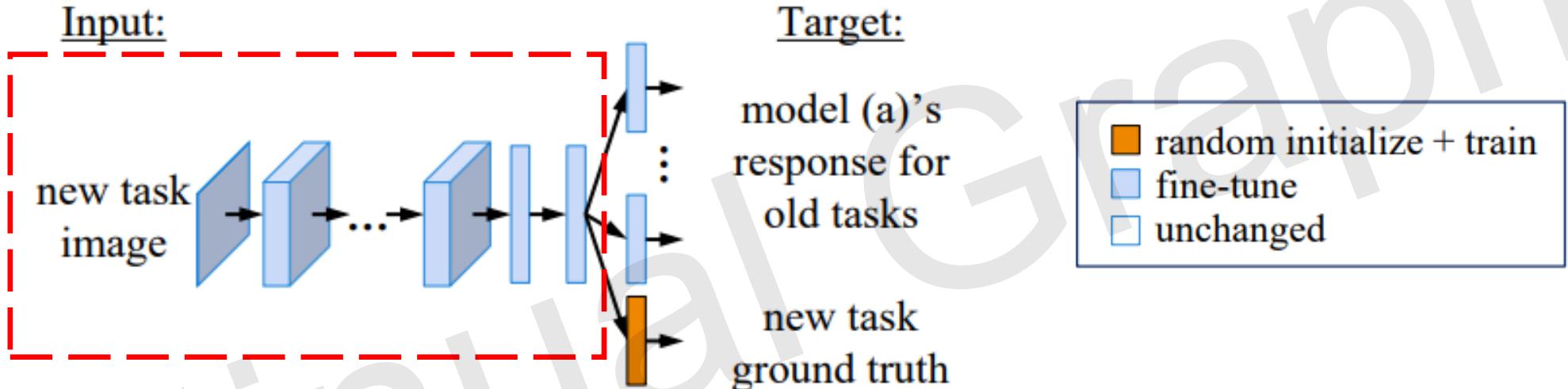
importance score (total)

$$\mathcal{L}'_{k+1}(W) = \mathcal{L}_{k+1}^{new}(W) + \sum_{n=1}^k I_n \otimes (W - W_n^*)^2$$

regularization

$$\mathcal{L}_{k+1}(W) = \mathcal{L}'_{k+1}(W) + \beta \|I_{k+1}\|_1$$

Learning without forgetting (LwF, regularization based)



Start with:

θ_s : shared parameters

θ_o : task specific parameters for each old task

X_n, Y_n : training data and ground truth on the new task

Initialize:

$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // compute output of old tasks for new data

$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // randomly initialize new parameters

Train:

Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // old task output

Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // new task output

$\theta_s^*, \theta_o^*, \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\text{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$

Self-Supervised Continual Graph Learning in Adaptive Riemannian Spaces (RieGrace, Regularization based)

RieGrace

Intra-distillation (self-supervised part): maximize the agreement between the final outputs and embeddings from shallow layer

$$\mathcal{J}(\mathbf{x}_i^{s,L}, \mathbf{x}_i^{s,H}) = -\log \frac{\exp \text{Sim}^{\mathcal{L}}(\mathbf{x}_i^{s,L}, \mathbf{x}_i^{s,H})}{\sum_{j=1}^{|\mathcal{V}|} \mathbb{I}\{i \neq j\} \exp \text{Sim}^{\mathcal{L}}(\mathbf{x}_i^{s,L}, \mathbf{x}_i^{s,H})}$$

Similarity between low- and high-level representations of node i

Inter-distillation: maximize the agreement between the outputs of old and new models

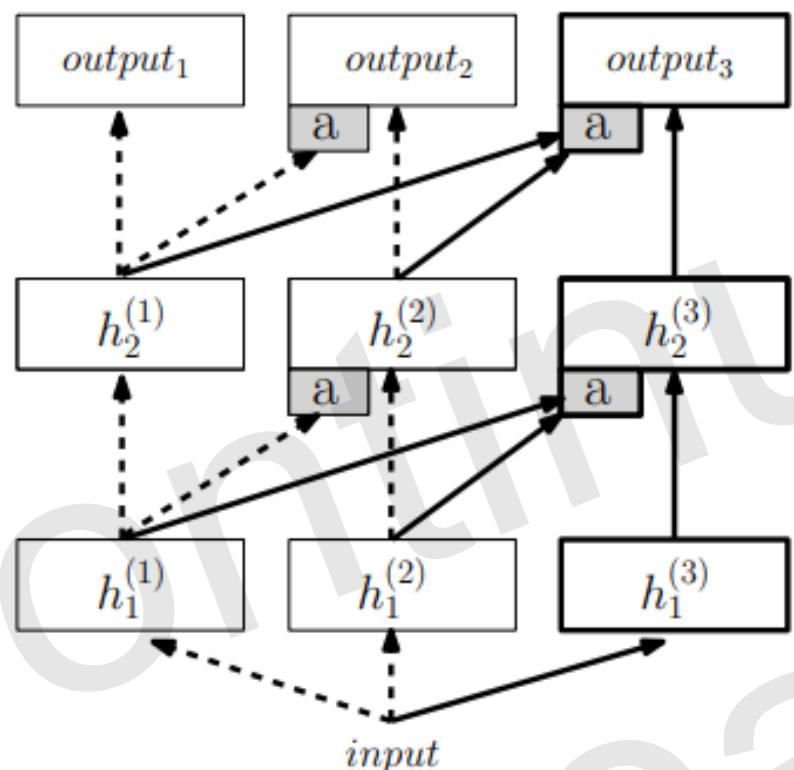
$$\mathcal{J}(\mathbf{x}_i^{t,H}, \mathbf{x}_i^{s,H}) = -\log \frac{\exp \text{Sim}^{\mathcal{L}}(\mathbf{x}_i^{t,H}, \mathbf{x}_i^{s,H})}{\sum_{j=1}^{|\mathcal{V}|} \mathbb{I}\{i \neq j\} \exp \text{Sim}^{\mathcal{L}}(\mathbf{x}_i^{t,H}, \mathbf{x}_i^{s,H})}$$

Similarity between high-level representations of node i from teacher (old) and student (new) model

Rough Categories

- **Regularization:** Penalize changes to model parameters via regularizations
- **Parameter-isolation:** Separate parameters for new and old tasks (partially or entirely)
- **Memory-replay:** Replay old task data to the model when learning new ones

Progressive neural network (Parameter isolation based)



Separate branch for each task

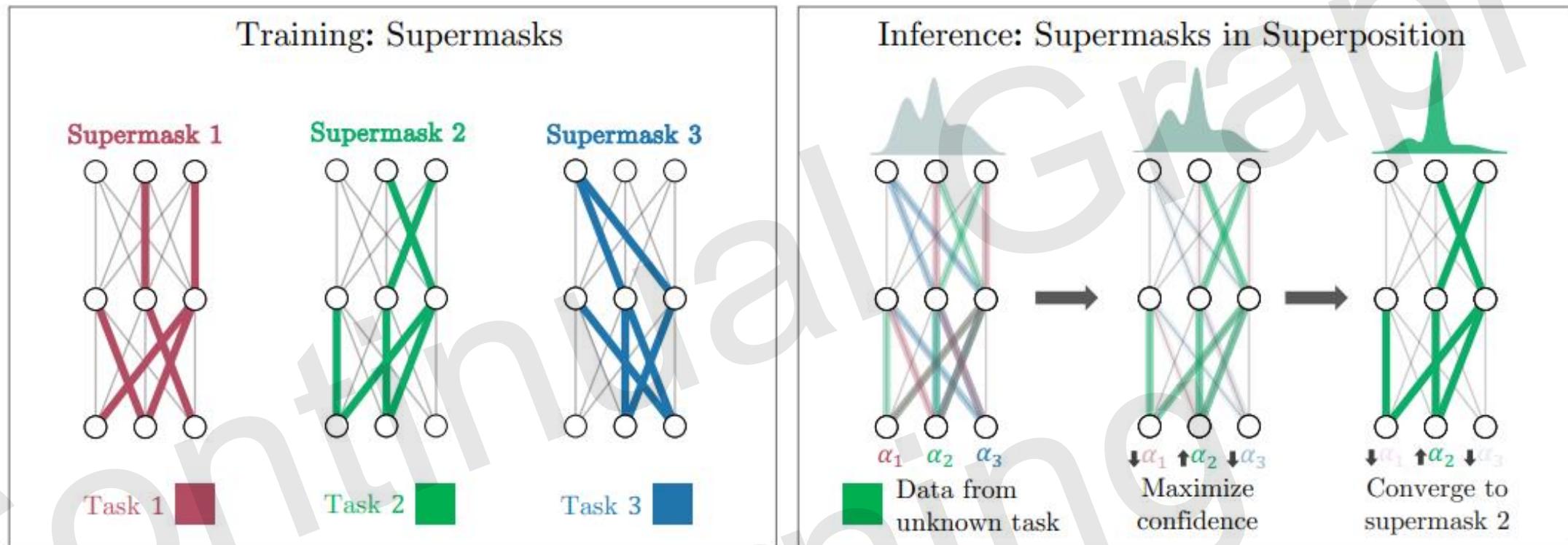
$$h_i^{(k)} = f \left(\boxed{W_i^{(k)} h_{i-1}^{(k)}} + \boxed{\sum_{j < k} U_i^{(k:j)} h_{i-1}^{(j)}} \right)$$

$W_i^{(k)}$ weight matrix of layer i of column k

$U_i^{(k:j)}$ layer $i - 1$ of column j , to layer i of column k

$$f(x) = \max(0, x)$$

Supermasks in Superposition (Parameter isolation based)



Learn a Supermask atop the backbone for each task:

$$\mathbf{p} = f(\mathbf{x}, \mathbf{W} \odot \mathbf{M})$$

Without task indicators:
$$\mathbf{p}(\alpha) = f\left(\mathbf{x}, \mathbf{W} \odot \left(\sum_{i=1}^k \alpha_i \mathbf{M}^i\right)\right)$$

$$\arg \max_i \left(-\frac{\partial \mathcal{H}(\mathbf{p}(\alpha))}{\partial \alpha_i} \right)$$

Hierarchical prototype networks (HPNs, parameter-isolation based)

Motivation: Inspired by human cognition process, Different representations of data in different classes may be better represented by different combinations of a shared pool of basic features

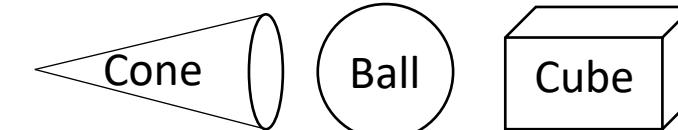
Basic feature types:

Color

Shape

Basic feature instances
(prototypes):

Orange Red Green



Objects (combinations of basic feature prototypes):



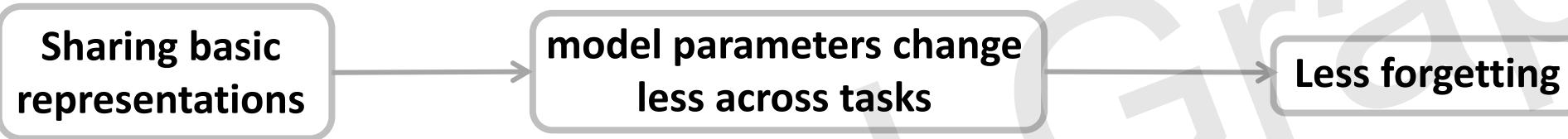
Class (higher level hierarchy):

Fruits

Vegetables

Hierarchical prototype networks (HPNs, parameter-isolation based)

Graph nodes (e.g. people in social network, papers in citation networks) should also be decomposable.



Analogy

Basic feature types
(color, shape)

Basic feature instances
(orange, red, green, etc.)

Objects (watermelon,
tomato, etc.)

Class (fruit, vegetable)

Our design

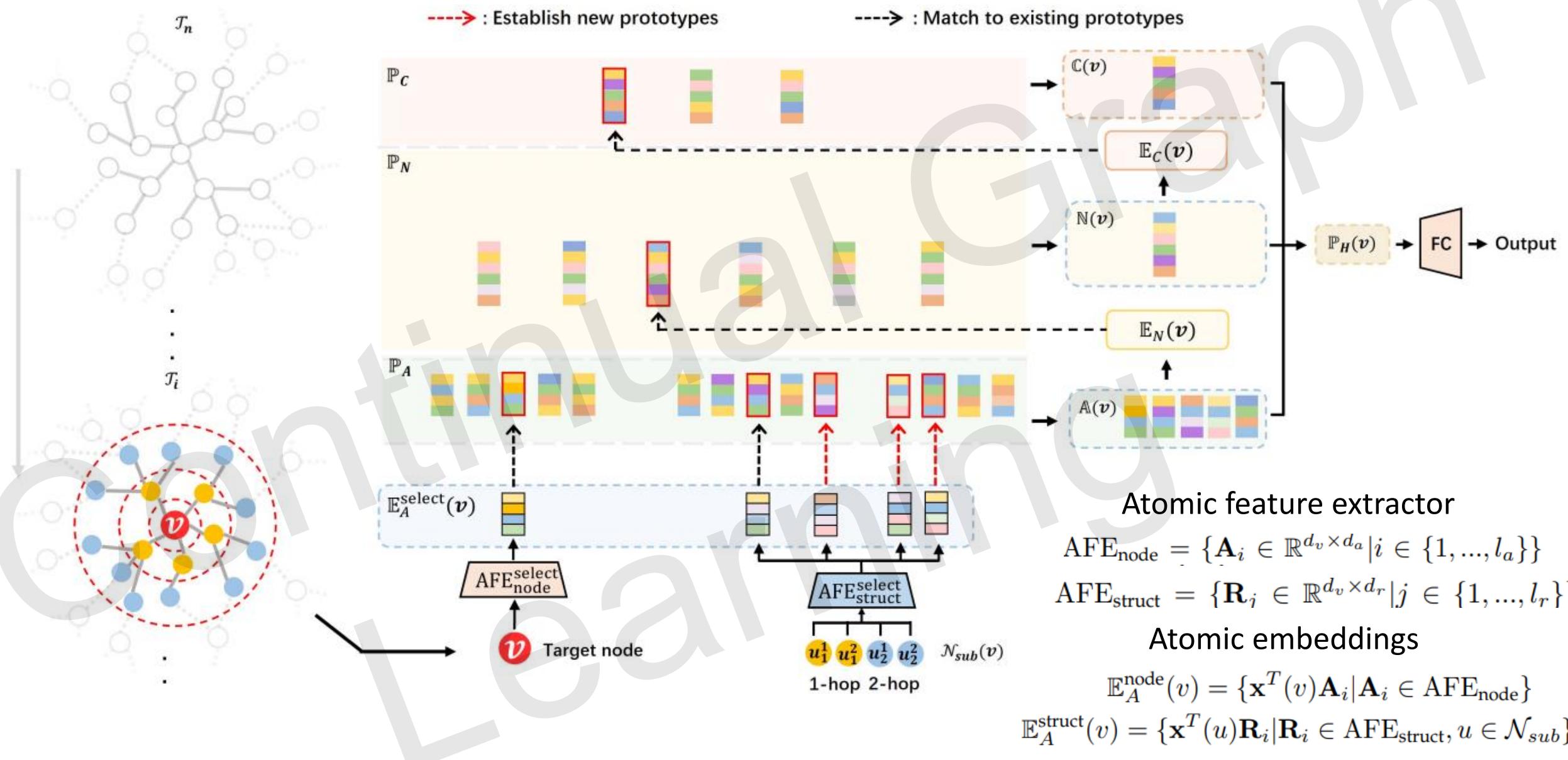
Atomic feature extractor
(AFE)

Atom prototype
(A-Prototype)

Node-level prototype
(N-Prototype)

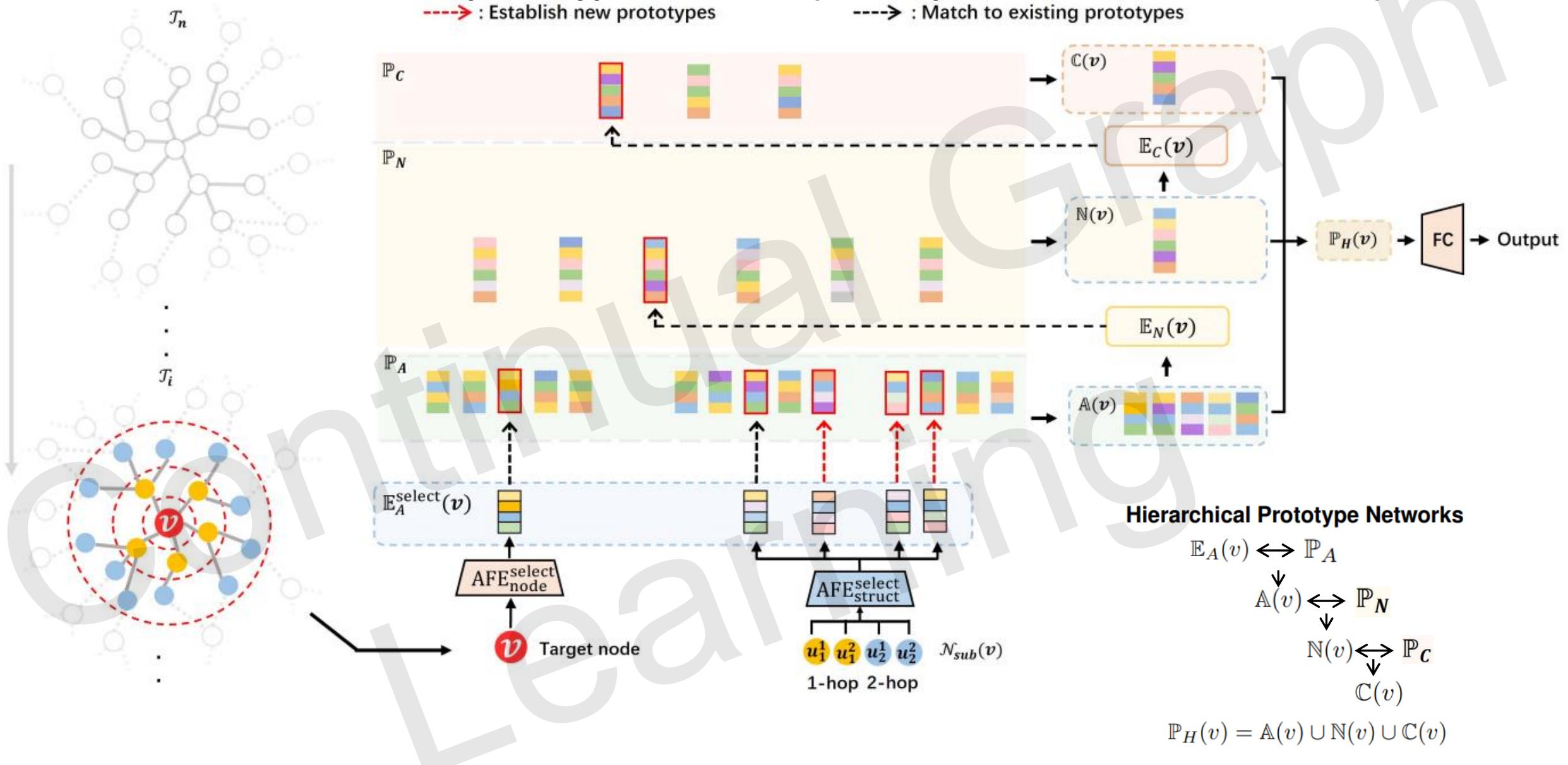
Class-level prototype
(C-Prototype)

Hierarchical prototype networks (HPNs, parameter-isolation based)



Zhang, Xikun, Dongjin Song, and Dacheng Tao. "Hierarchical prototype networks for continual graph representation learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

Hierarchical prototype networks (HPNs, parameter-isolation based)



- Theoretical justification of the capability to overcome forgetting

Theorem 2 (Task distance preserving). *For HPNs trained on consecutive tasks \mathcal{T}^p and \mathcal{T}^{p+1} . If $l_a d_a + l_r d_r \geq (l_r + 1)d_v$ and \mathbf{W} is column full rank, then as long as $t_A < \lambda_{\min}(l_r + 1)\text{dist}(\mathbb{V}_p, \mathbb{V}_{p+1})$, learning on \mathcal{T}^{p+1} will not modify representations HPNs generate for data from \mathcal{T}^p , i.e., catastrophic forgetting is avoided.*

Hierarchical prototype networks (HPNs, parameter-isolation based)

C.L.T.	Base	Cora		Citeseer		Actor		OGB-Arxiv		OGB-Products	
		AM/%	FM/%	AM/%	FM/%	AM/%	FM /%	AM/%	FM /%	AM/%	FM /%
None	GCN	63.5±1.9	-42.3±0.4	64.5±3.9	-7.7±1.6	43.6±3.6	-9.1±2.9	56.8±4.3	-19.8±3.2	45.2±5.6	-27.8±7.1
	GAT	71.9±3.8	-33.1±2.3	66.8±0.9	-19.6±0.3	53.1±2.7	-4.3±1.6	54.3±3.5	-21.7±4.6	44.9±6.9	-30.3±5.2
	GIN	68.3±2.3	-35.4±3.4	57.7±2.3	-36.4±0.3	45.5±2.3	-8.9±2.8	53.2±6.5	-23.5±8.1	43.1±7.4	-31.4±8.8
EWC [2]	GCN	63.1±1.2	-42.7±1.6	54.4±4.2	-30.3±0.9	44.3±1.1	-7.1±1.4	72.1±2.4	-9.1±1.9	66.7±0.5	-8.4±0.4
	GAT	72.2±1.5	-32.2±1.6	65.7±2.5	-19.7±2.3	54.2±2.5	-2.5±1.5	73.2±1.1	-10.8±2.1	67.9±1.0	-9.6±1.3
	GIN	69.6±2.6	-28.5±2.8	57.9±3.4	-36.3±2.4	47.6±2.1	-7.2±1.6	74.1±1.7	-8.3±2.0	67.3±2.3	-13.6±1.5
LwF [23]	GCN	76.1±1.4	-21.3±2.4	67.0±0.2	-8.3±2.7	49.7±2.5	-3.6±1.4	69.9±3.9	-12.1±2.8	66.3±2.5	-11.8±3.4
	GAT	70.8±2.8	-34.6±4.1	66.1±4.1	-18.9±1.5	52.8±2.7	-6.2±2.2	68.9±4.4	-13.6±3.3	65.1±4.1	-13.2±2.9
	GIN	74.1±2.7	-23.3±0.8	63.1±1.9	-16.5±2.2	49.7±2.6	-4.1±1.5	71.4±4.8	-15.9±5.6	65.9±4.0	-10.7±3.1
GEM [2]	GCN	75.7±3.0	-6.5±4.4	41.8±2.6	-31.9±1.4	52.7±3.1	+3.9±2.9	75.4±1.7	-13.6±0.5	71.3±1.7	-10.5±0.9
	GAT	69.8±3.0	-26.1±2.6	71.3±2.2	+9.0±1.5	54.3±3.6	-2.0±0.9	76.6±0.7	-11.3±0.4	70.4±0.8	-10.9±1.6
	GIN	80.2±3.3	-2.0±4.2	49.7±0.5	-24.5±0.9	45.2±2.8	-11.1±1.5	77.3±2.1	-11.2±1.6	76.5±3.3	-7.2±2.5
MAS [59]	GCN	65.5±1.9	-21.4±3.7	59.5±3.1	-0.1±2.4	50.7±2.4	-1.5±0.8	69.8±0.4	-18.8±0.9	62.0±1.1	-17.9±1.9
	GAT	84.7±0.7	-5.6±2.0	69.1±1.1	-4.8±3.3	53.7±2.6	-1.6±0.8	70.6±1.3	-16.7±1.6	64.4±2.3	-14.5±3.2
	GIN	76.7±2.6	-4.0±3.6	65.2±3.9	+0.0±2.0	51.6±2.6	-0.6±1.3	65.3±2.9	-17.0±2.3	61.4±3.8	-20.9±2.9
ERGN. [60]	GCN	63.5±2.4	-42.3±0.7	54.2±3.9	-30.3±1.9	52.4±3.3	+0.6±1.4	63.3±1.7	-18.1±0.9	60.7±2.8	-26.6±3.3
	GAT	71.1±2.5	-34.3±1.0	65.5±0.3	-20.4±3.9	51.4±2.2	-7.2±3.2	63.5±2.4	-19.5±1.9	61.3±1.7	-25.1±0.8
	GIN	68.3±0.4	-35.4±0.4	57.7±3.1	-36.4±1.3	42.7±3.9	-13.0±2.1	69.2±1.8	-11.8±1.4	61.8±4.7	-23.4±7.9
TWP [18]	GCN	68.9±0.9	-5.7±1.5	60.5±3.8	-0.3±4.4	50.6±2.0	-4.8±1.1	75.6±0.3	-10.4±0.5	69.9±0.4	-9.0±1.1
	GAT	81.3±3.2	-14.4±1.5	69.8±1.5	-8.9±2.6	54.0±1.8	-2.1±1.9	75.8±0.5	-5.9±0.3	69.3±2.3	-8.9±1.5
	GIN	73.7±3.2	-3.9 ±2.6	68.9±0.7	-2.4±1.9	49.9±1.9	-3.6±2.0	76.6±1.8	-11.3±1.1	69.9±1.4	-10.3±2.7
Join.	GCN	93.7± 0.5	-	78.9±0.4	-	57.0±0.9	-	77.2±0.8	-	72.9±1.2	-
	GAT	93.9± 0.9	-	79.3±0.8	-	57.1±0.9	-	81.8±0.3	-	73.7±2.4	-
	GIN	93.2± 1.2	-	78.7±0.9	-	56.9±0.6	-	82.3±1.9	-	77.9±2.1	-
HPNs		93.7±1.5	+0.6±1.0	79.0±0.9	-0.6±0.7	56.8±1.4	-0.9±0.9	85.8± 0.7	+0.6±0.9	80.1±0.8	+2.9±1.0

Comparison with State-of-the-arts

Hierarchical prototype networks (HPNs, parameter-isolation based)

- Theoretical upper bound on the number of prototypes

Theorem 1 (Upper bounds on numbers of prototypes). *Given the notations defined in HPNs, the upper bound on the number of A-prototypes n_a can be given by*

$$n_A \leq (l_a + l_r) \max_N S(d_a, N, 1 - t_A), \quad (14)$$

and the upper bounds on the number of N-prototypes and the C-prototypes are:

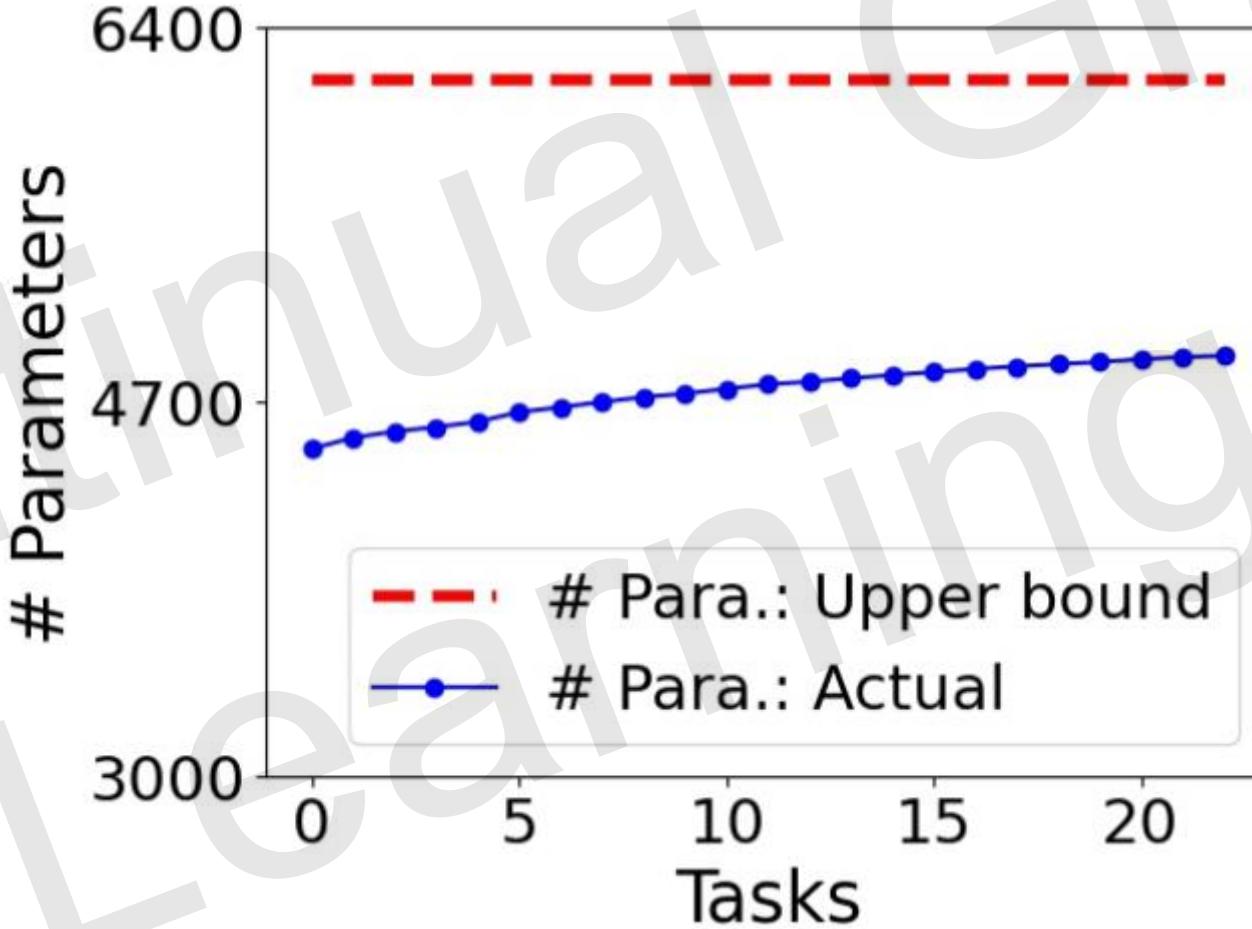
$$n_N \leq \max_N S(d_n, N, 1 - t_N) \quad (15)$$

$$n_C \leq \max_N S(d_c, N, 1 - t_C) \quad (16)$$

where $S(n, N, t)$ is the spherical code defined on a n dimensional hypersphere.

Hierarchical prototype networks (HPNs, parameter-isolation based)

- Theoretical upper bound on the number of prototypes



Rough Categories

- **Regularization:** Penalize changes to model parameters via regularizations
- **Parameter-isolation:** Separate parameters for new and old tasks (partially or entirely)
- **Memory-replay:** Replay old task data to the model when learning new ones

Tiny Episodic Memories (memory replay based)

Algorithm 1 Experience Replay for Continual Learning.

```
1: procedure ER( $\mathcal{D}$ , mem_sz, batch_sz, lr)
2:    $\mathcal{M} \leftarrow \{\} * \text{mem\_sz}$                                  $\triangleright$  Allocate memory buffer of size mem_sz
3:    $n \leftarrow 0$                                                   $\triangleright$  Number of training examples seen in the continuum
4:   for  $t \in \{1, \dots, T\}$  do
5:     for  $B_n \stackrel{K}{\sim} \mathcal{D}_t$  do                                 $\triangleright$  Sample without replacement a mini-batch of size  $K$  from task  $t$ 
6:        $B_{\mathcal{M}} \stackrel{K}{\sim} \mathcal{M}$                                           $\triangleright$  Sample a mini-batch from  $\mathcal{M}$ 
7:        $\theta \leftarrow SGD(B_n \cup B_{\mathcal{M}}, \theta, lr)$                           $\triangleright$  Single gradient step to update the parameters by stacking current minibatch with
minibatch from memory
8:        $\mathcal{M} \leftarrow \text{UpdateMemory(mem\_sz, } t, n, B_n)$                    $\triangleright$  Memory update, see §4
9:        $n \leftarrow n + \text{batch\_sz}$                                           $\triangleright$  Counter update
10:  return  $\theta, \mathcal{M}$                                               Memory update
```

Tiny Episodic Memories (memory replay based)

Algorithm 2 Reservoir sampling update. mem_sz is the number of examples the memory can store, t is the task id, n is the number of examples observed so far in the data stream, and B is the input mini-batch.

```
1: procedure UPDATEREMORY( $\text{mem\_sz}, t, n, B$ )
2:    $j \leftarrow 0$ 
3:   for  $(\mathbf{x}, y)$  in  $B$  do
4:      $M \leftarrow |\mathcal{M}|$ 
5:     if  $M < \text{mem\_sz}$  then
6:        $\mathcal{M}.\text{append}(\mathbf{x}, y, t)$ 
7:     else
8:        $i = \text{randint}(0, n + j)$ 
9:       if  $i < \text{mem\_sz}$  then
10:         $\mathcal{M}[i] \leftarrow (\mathbf{x}, y, t)$ 
11:     $j \leftarrow j + 1$ 
12: return  $\mathcal{M}$ 
```

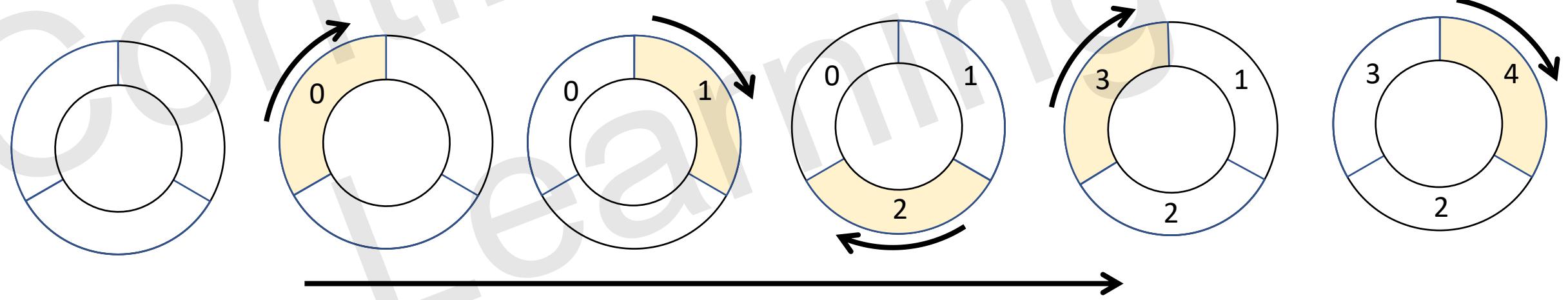
▷ Number of samples currently stored in the memory

▷ Overwrite memory slot.

Tiny Episodic Memories (memory replay based)

Algorithm 3 Ring buffer.

```
1: procedure UPDATERMEMORY(mem_sz,  $t$ ,  $n$ ,  $B$ )
2:   for  $(x, y)$  in  $B$  do
3:     # Assume FIFO stacks  $\mathcal{M}[t][y]$  of fixed size are already initialized
4:      $\mathcal{M}[t][y].append(x)$ 
5:   return  $\mathcal{M}$ 
```



Tiny Episodic Memories (memory replay based)

Algorithm 4 K-Means. Memory is populated using samples closest (in feature space) to sequential K-Means centroids.

```
1: procedure UPDATERMEMORY(mem_sz, t, n, B)
2:   # Assume array  $\mathcal{M}[t][y]$  of fixed size is already initialized
3:   # Assume K centroids  $c_j$  are already initialized
4:   # Assume cluster counters  $n_j$  are already initialized to 0
5:   for  $(\mathbf{x}, y)$  in  $B$  do
6:      $j \leftarrow \operatorname{argmin}_{j \in \{1, \dots, K\}} \|\phi_\theta(\mathbf{x}) - c_j\|$       Choose the closest cluster
7:      $n_j \leftarrow n_j + 1$ 
8:      $c_j \leftarrow c_j + \frac{1}{n_j} * (\phi_\theta(\mathbf{x}) - c_j)$           Update the cluster center
9:      $d = \|\phi_\theta(\mathbf{x}) - c_j\|$ 
10:    if  $d < \mathcal{M}[t][y][j].get\_dst()$  then
11:       $\mathcal{M}[t][y][j] \leftarrow (\mathbf{x}, d)$                                 ▷ Store the current example if it is closer to the centroid
12:   return  $\mathcal{M}$ 
```

Tiny Episodic Memories (memory replay based)

Algorithm 5 Mean of Features. Store examples that are closest to the running average feature vector.

```
1: procedure UPDATERMEMORY(mem_sz, t, n, B)
2:   # Assume heaps  $\mathcal{M}[t][y]$  of fixed size are already initialized
3:   # Assume average features  $f[t][y]$  are already initialized
4:   # Assume moving average decay hyper-parameter ( $\alpha$ ) is given
5:   for  $(\mathbf{x}, y)$  in  $B$  do
6:      $f[t][y] \leftarrow \alpha * f[t][y] + (1 - \alpha) * \phi_\theta(\mathbf{x})$            Update the running average
7:      $d = \|\phi_\theta(\mathbf{x}) - f[t][y]\|$ 
8:     if  $\mathcal{M}[t][y].\text{find\_max}() > d$  then                                ▷ Store the current example if it is closer to the center
9:        $\mathcal{M}[t][y].\text{delete\_max}()$ 
10:       $\mathcal{M}[t][y].\text{insert}(\mathbf{x}; d)$ 
11:   return  $\mathcal{M}$ 
```

Gradient episodic memory (GEM, memory replay based)

Loss of the stored data:

$$\ell(f_\theta, \mathcal{M}_k) = \frac{1}{|\mathcal{M}_k|} \sum_{(x_i, k, y_i) \in \mathcal{M}_k} \ell(f_\theta(x_i, k), y_i)$$

Memory buffer storing
representative data from an
old task k

When observing a new triplet (x, y, t) :

$$\begin{aligned} & \text{minimize}_\theta \quad \ell(f_\theta(x, t), y) \\ & \text{subject to} \quad \ell(f_\theta, \mathcal{M}_k) \leq \ell(f_\theta^{t-1}, \mathcal{M}_k) \text{ for all } k < t \end{aligned}$$

$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x, t), y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t.$$

Gradient episodic memory (GEM, memory replay based)

Loss of the stored data:

$$\ell(f_\theta, \mathcal{M}_k) = \frac{1}{|\mathcal{M}_k|} \sum_{(x_i, k, y_i) \in \mathcal{M}_k} \ell(f_\theta(x_i, k), y_i)$$

Memory buffer storing
representative data from an
old task k

When observing a new triplet (x, y, t) :

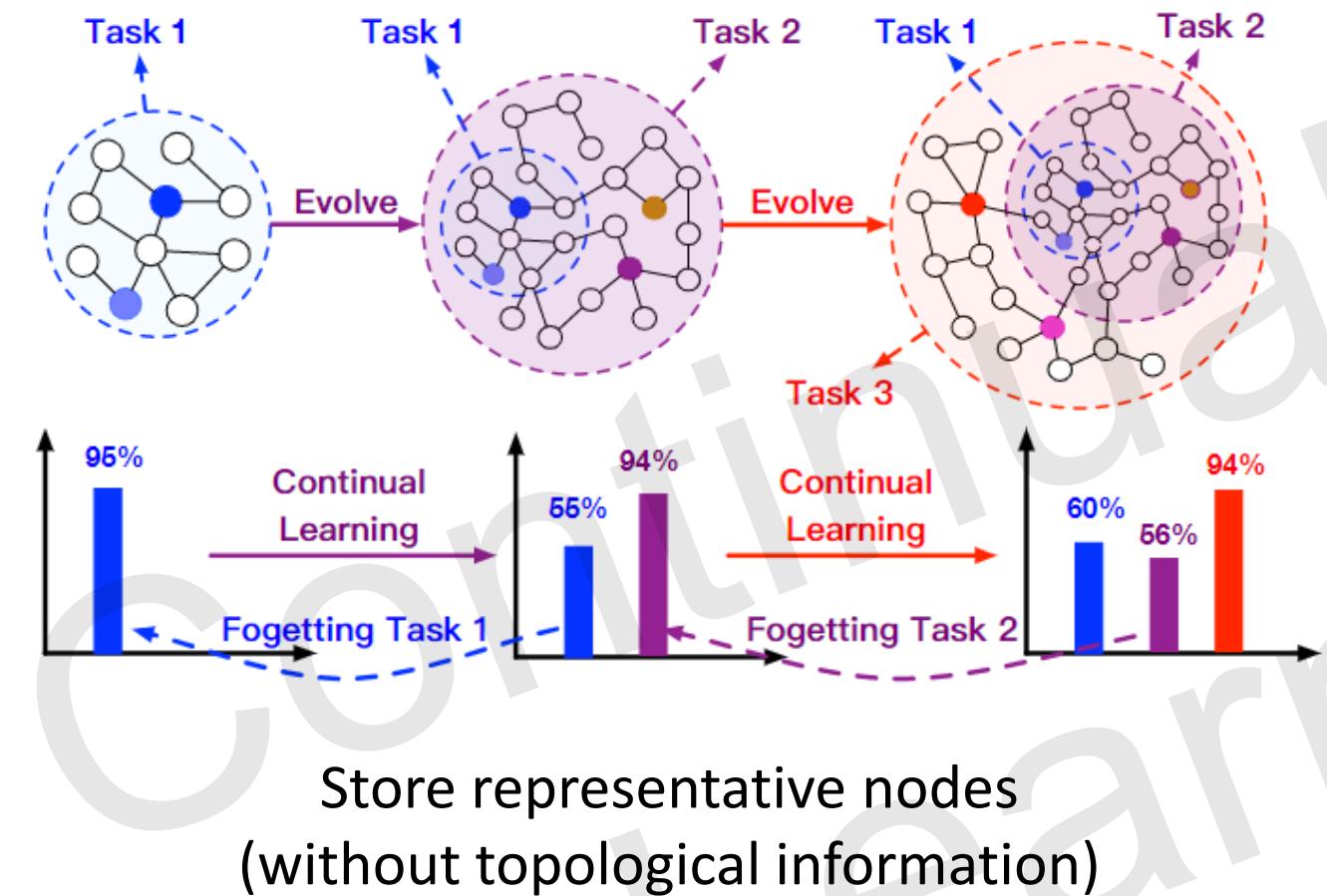
$$\text{minimize}_\theta \quad \ell(f_\theta(x, t), y)$$

$$\text{subject to} \quad \ell(f_\theta, \mathcal{M}_k) \leq \ell(f_\theta^{t-1}, \mathcal{M}_k) \text{ for all } k < t$$

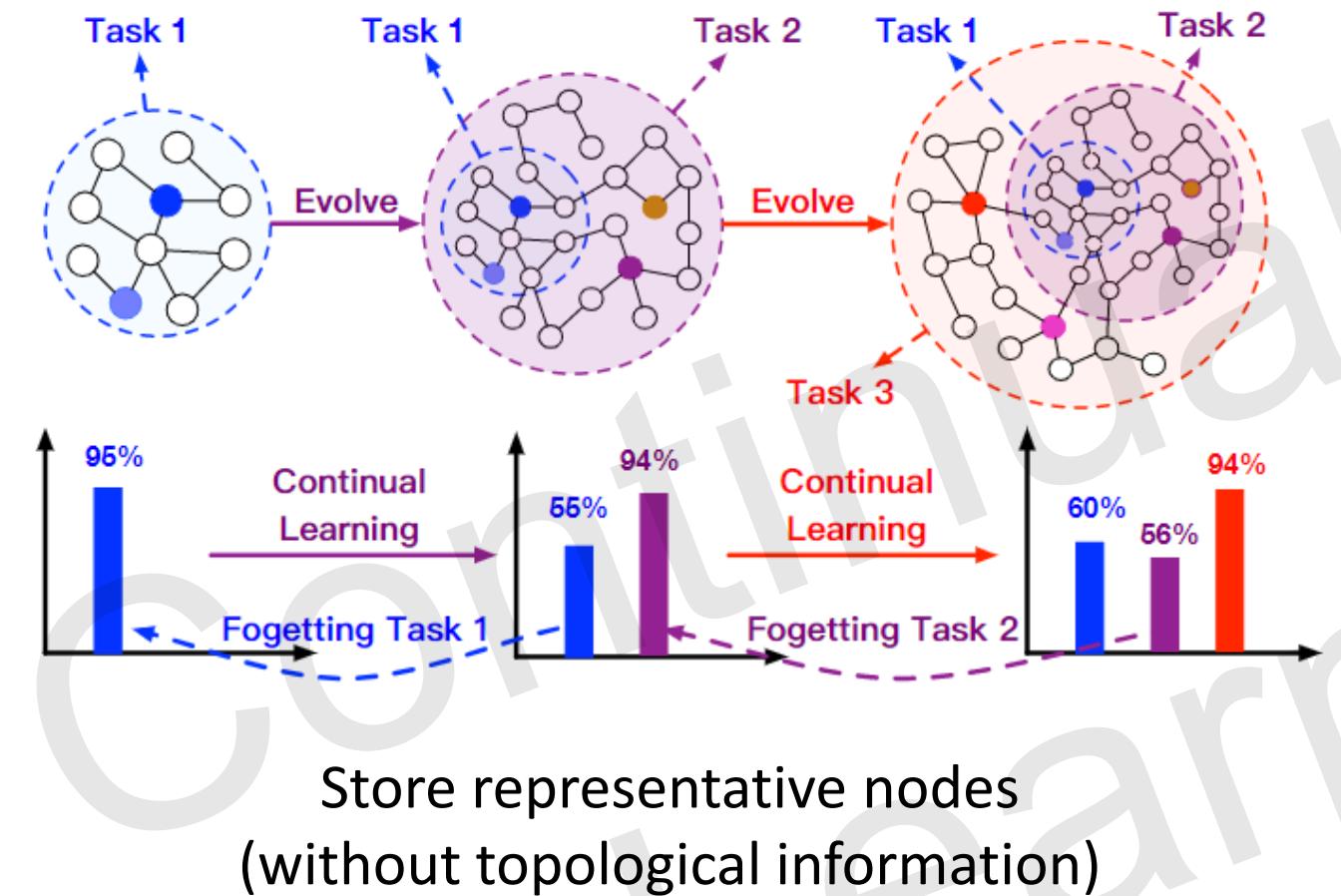
$$\langle g, g_k \rangle := \left\langle \frac{\partial \ell(f_\theta(x, t), y)}{\partial \theta}, \frac{\partial \ell(f_\theta, \mathcal{M}_k)}{\partial \theta} \right\rangle \geq 0, \text{ for all } k < t.$$

Aim: Find directions in the parameter space that are favourable for all tasks

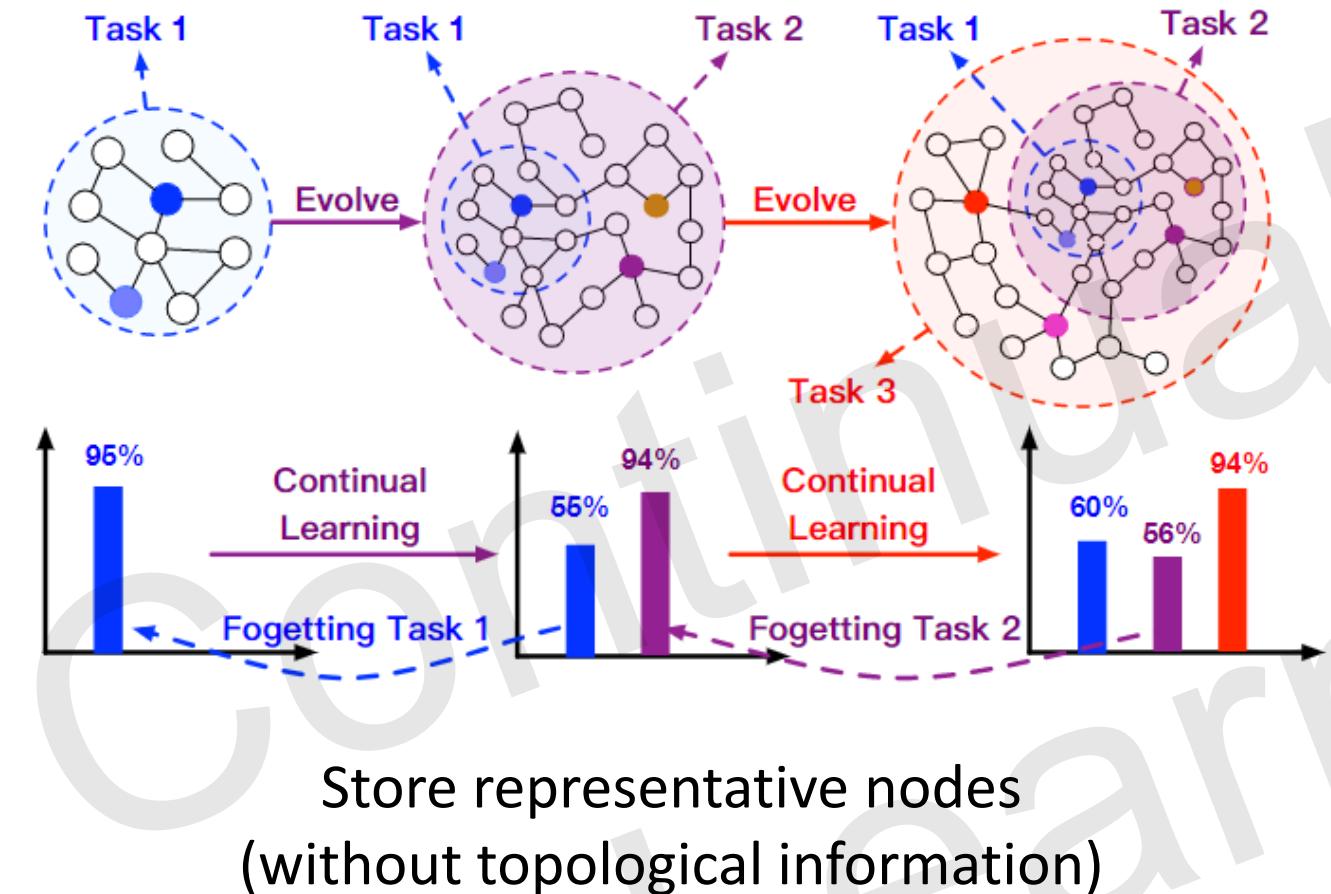
Experience Replay Graph Neural Networks (ER-GNN, memory replay based)



Experience Replay Graph Neural Networks (ER-GNN, memory replay based)



Experience Replay Graph Neural Networks (ER-GNN, memory replay based)



Original loss:

$$\mathcal{L}_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}) = - \left(\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} (y_i \log f_{\theta}(\mathbf{x}_i) + (1 - y_i) \log(1 - f_{\theta}(\mathbf{x}_i))) \right)$$

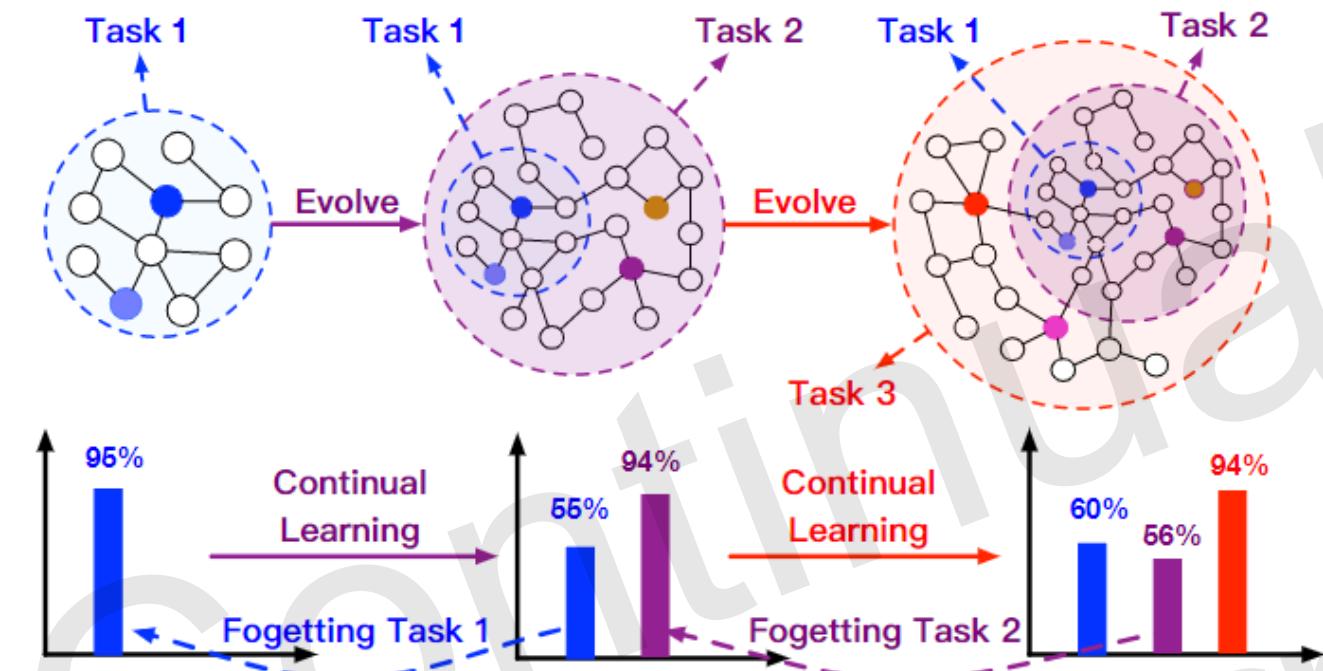
Loss with memory buffer:

$$\mathcal{L}'_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}_i^{\text{tr}}, B) = \beta \mathcal{L}_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}_i^{\text{tr}}) + (1 - \beta) \mathcal{L}_{\mathcal{T}_i}(f_{\theta}, B)$$

Objective:

$$\theta = \arg \min_{\theta \in \Theta} (\mathcal{L}'_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}_i^{\text{tr}}, B))$$

Experience Replay Graph Neural Networks (ER-GNN, memory replay based)



Store representative nodes
(without topological information)

Original loss:

$$\mathcal{L}_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}) = -(\sum_{(x_i, y_i) \in \mathcal{D}} (y_i \log f_{\theta}(x_i) + (1 - y_i) \log(1 - f_{\theta}(x_i))))$$

Loss with memory buffer:

$$\mathcal{L}'_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}_i^{\text{tr}}, B) = \beta \mathcal{L}_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}_i^{\text{tr}}) + (1 - \beta) \mathcal{L}_{\mathcal{T}_i}(f_{\theta}, B)$$

Objective:

$$\theta = \arg \min_{\theta \in \Theta} (\mathcal{L}'_{\mathcal{T}_i}(f_{\theta}, \mathcal{D}_i^{\text{tr}}, B))$$

Balance the contribution:

$$\beta = |B| / (|\mathcal{D}_i^{\text{tr}}| + |B|)$$

$|\mathcal{D}_i^{\text{tr}}|$: Size of training set of the current task

$|B|$: Size of training set of the memory buffer

Sparsified Subgraph Memory (SSM, memory replay based)

GNNs generate node representations based on multi-hop neighborhood



Applying memory replay to graphs requires storing multi-hop neighbors



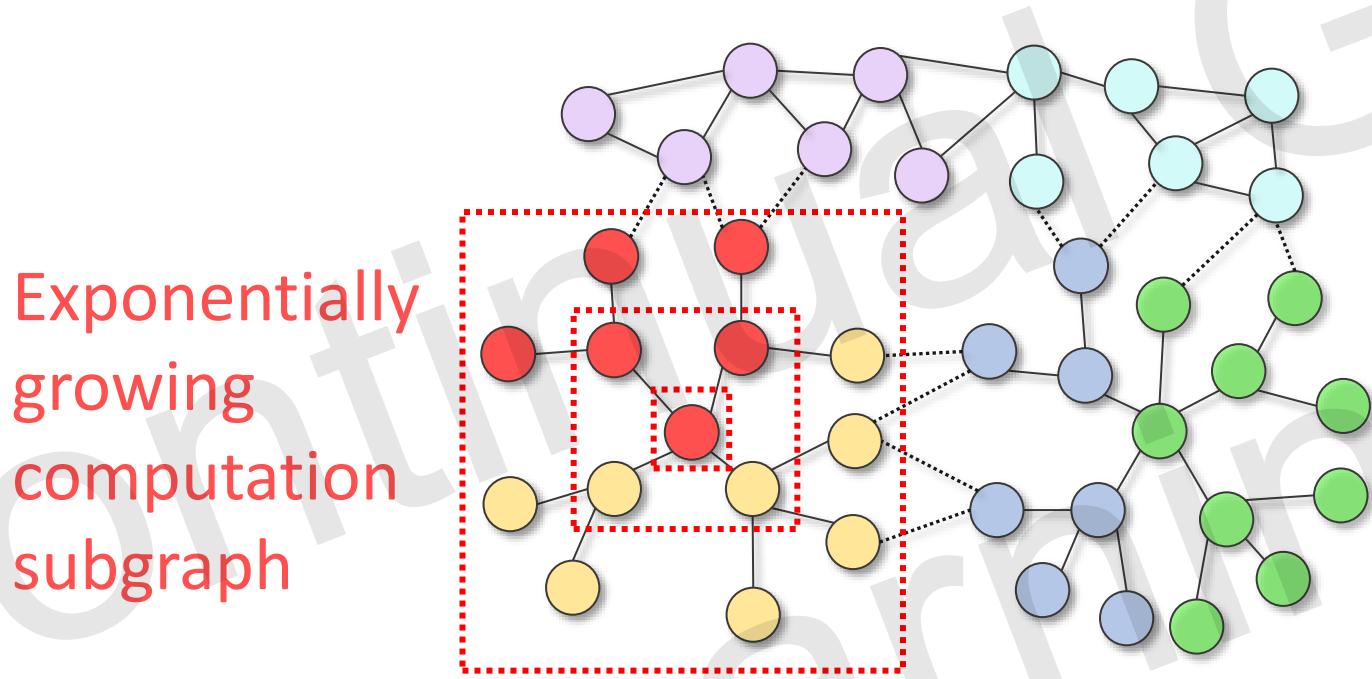
L-hop neighbors grow in $O(d^L)$, d is average degree



Memory explosion problem

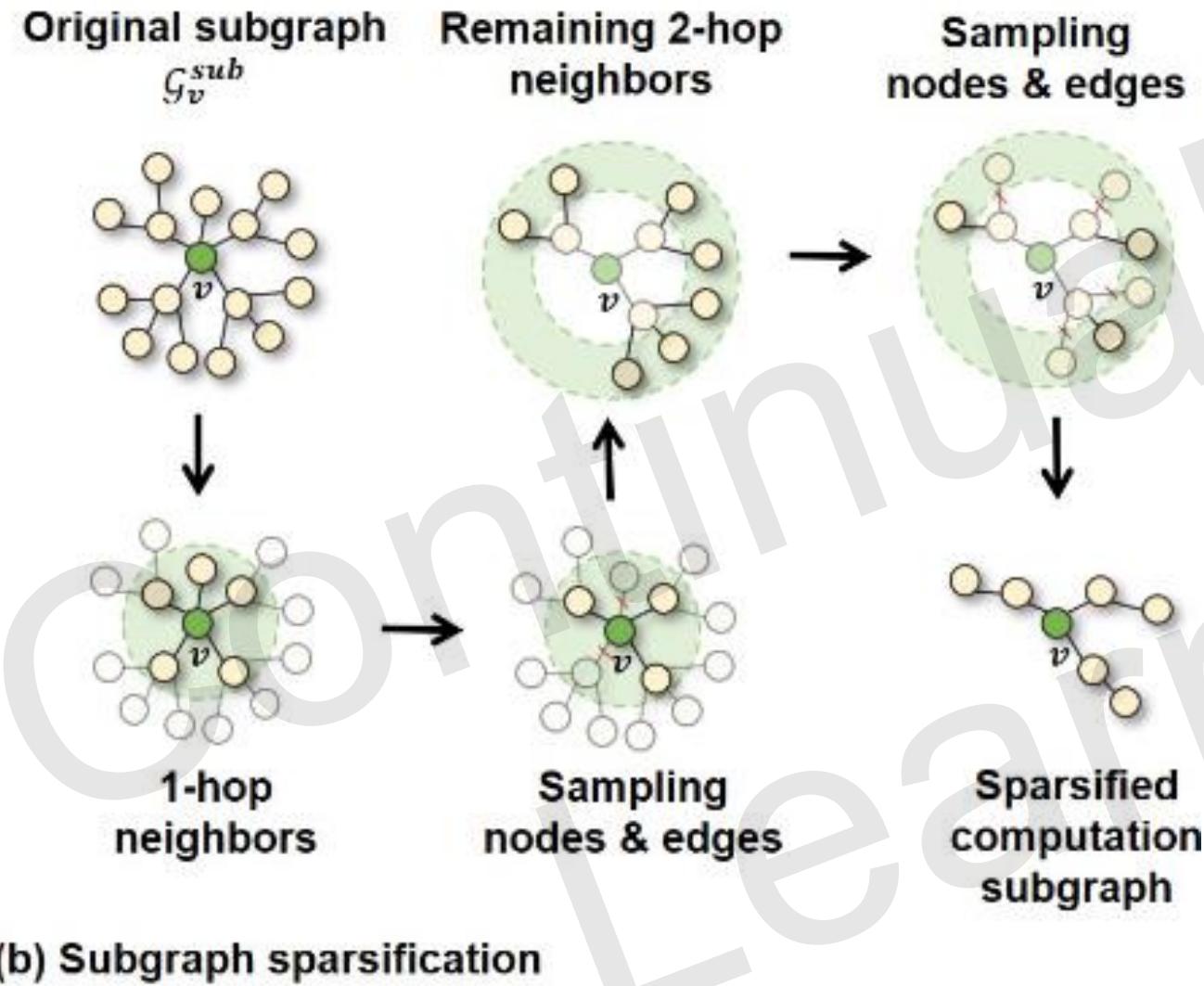
Sparsified Subgraph Memory (SSM, memory replay based)

- GNNs typically require no less than 2-hop computation subgraphs (i.e. at least two layers)



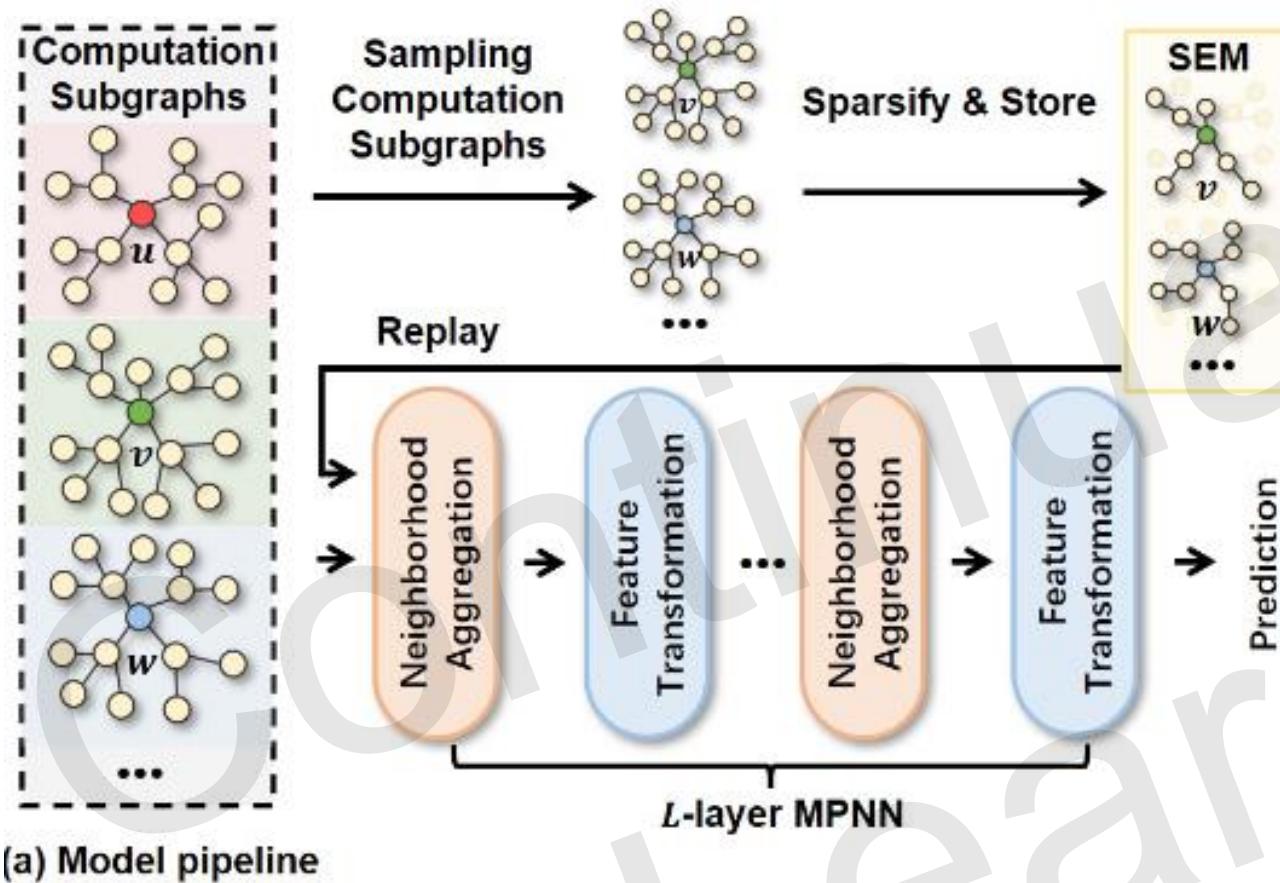
- Size of a computation subgraph grows in $O(d^L)$, d is average degree, L is number of hops
- Memory explosion (e.g. $d = 492$ in Reddit dataset)

Sparsified Subgraph Memory (SSM, memory replay based)



- Sparsify hop by hop
- First selected a fixed number of 1-hop neighbors
- Then selected a fixed number of 2-hop neighbors (must be connected to selected 1-hop neighbors)
- Any sampling strategy can be used

Sparsified Subgraph Memory (SSM, memory replay based)



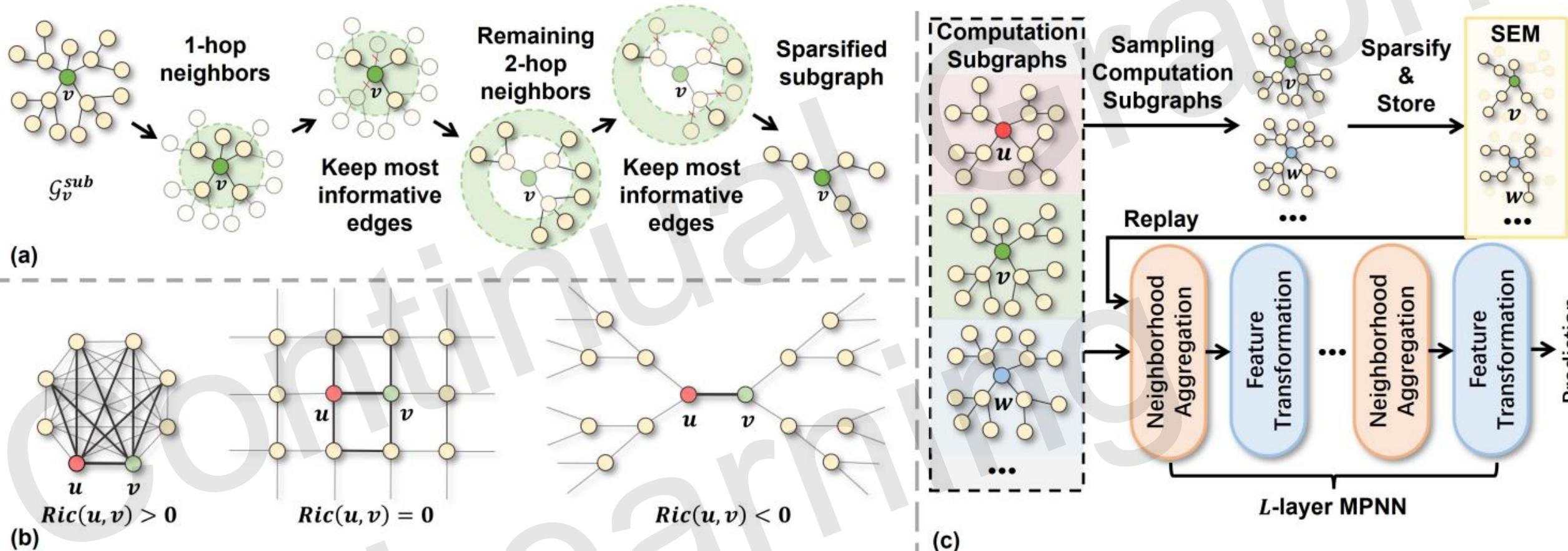
- Each node has a computation subgraph, size is $O(d^L)$
- Selected a fixed number of computations to store
- Sparsify each selected computation subgraph into a fixed number of nodes, size reduces to $O(1)$
- Plug and play, easy to use in any GNNs

Sparsified Subgraph Memory (SSM, memory replay based)

Continual learning techniques	CoraFull		OGB-Arxiv		Reddit		OGB-Products	
	AA/% ↑	AF/% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑
Fine-tune	3.5±0.5	-95.2±0.5	4.9±0.0	-89.7±0.4	5.9±1.2	-97.9±3.3	3.4±0.8	-82.5±0.8
EWC [19]	52.6±8.2	-38.5±12.1	8.5±1.0	-69.5±8.0	10.3±11.6	-33.2±26.1	23.8±3.8	-21.7±7.5
MAS [21]	12.3±3.8	-83.7±4.1	4.9±0.0	-86.8±0.6	13.1±2.6	-35.2±3.5	16.7±4.8	-57.0±31.9
GEM [7]	8.4±1.1	-88.4±1.4	4.9±0.0	-89.8±0.3	28.4±3.5	-71.9±4.2	5.5±0.7	-84.3±0.9
TWP [27]	62.6±2.2	-30.6±4.3	6.7±1.5	-50.6±13.2	13.5±2.6	-89.7±2.7	14.1±4.0	-11.4±2.0
LwF [20]	33.4±1.6	-59.6±2.2	9.9±12.1	-43.6±11.9	86.6±1.1	-9.2±1.1	48.2±1.6	-18.6±1.6
ER-GNN [9]	34.5±4.4	-61.6±4.3	30.3±1.5	-54.0±1.3	88.5±2.3	-10.8±2.4	56.7±0.3	-33.3±0.5
Joint (Not under continual setting)	81.2±0.4	-3.3±0.8	51.3±0.5	-6.7±0.5	97.1±0.1	-0.7±0.1	71.5±0.1	-5.8±0.3
SEM-uniform (Ours)	73.0±0.3	-14.8±0.5	47.1±0.5	-11.7±1.5	94.3±0.1	-1.4±0.1	62.0±1.6	-9.9±1.3
SEM-degree (Ours)	75.4±0.1	-9.7±0.0	48.3±0.5	-10.7±0.3	94.4±0.0	-1.3±0.0	63.3±0.1	-9.6±0.3

Comparison under the challenging class-II

Ricci Curvature-Based Graph Sparsification



Ricci Curvature-Based Graph Sparsification

Continual learning techniques	CoraFull		OGB-Arxiv		Reddit		OGB-Products	
	AA/% ↑	AF/% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑
Fine-tune	3.5±0.5	-95.2±0.5	4.9±0.0	-89.7±0.4	5.9±1.2	-97.9±3.3	7.6±0.7	-88.7±0.8
EWC [25]	52.6±8.2	-38.5±12.1	8.5±1.0	-69.5±8.0	10.3±11.6	-33.2±26.1	23.8±3.8	-21.7±7.5
MAS [27]	6.5±1.5	-92.3±1.5	4.8±0.4	-72.2±4.1	9.2±14.5	-23.1±28.2	16.7±4.8	-57.0±31.9
GEM [7]	8.4±1.1	-88.4±1.4	4.9±0.0	-89.8±0.3	11.5±5.5	-92.4±5.9	4.5±1.3	-94.7±0.4
TWP [38]	62.6±2.2	-30.6±4.3	6.7±1.5	-50.6±13.2	8.0±5.2	-18.8±9.0	14.1±4.0	-11.4±2.0
LwF [26]	33.4±1.6	-59.6±2.2	9.9±12.1	-43.6±11.9	86.6±1.1	-9.2±1.1	48.2±1.6	-18.6±1.6
ER-GNN [9]	34.5±4.4	-61.6±4.3	21.5±5.4	-70.0±5.5	82.7±0.4	-17.3±0.4	48.3±1.2	-45.7±1.3
SSM-uniform [23]	73.0±0.3	-14.8±0.5	47.1±0.5	-11.7±1.5	94.3±0.1	-1.4±0.1	62.0±1.6	-9.9±1.3
SSM-degree [23]	75.4±0.1	-9.7±0.0	48.3±0.5	-10.7±0.3	94.4±0.0	-1.3±0.0	63.3±0.1	-9.6±0.3
Joint (Not under continual setting)	81.2±0.4	-	51.3±0.5	-	97.1±0.1	-	71.5±0.1	-
SEM-curvature (Ours)	77.7±0.8	-10.0±1.2	49.9±0.6	-8.4±1.3	96.3±0.1	-0.6±0.1	65.1±1.0	-9.5±0.8

Rough Categories

- **Regularization:** Penalize changes to model parameters via regularizations
- **Parameter-isolation:** Separate parameters for new and old tasks (partially or entirely)
- **Memory-replay:** Replay old task data to the model when learning new ones

Pros:

- Good stability on old tasks
- Suitable for different architectures
- Good stability and plasticity
- Good stability and plasticity
- Suitable for different architectures

Cons:

- Poor plasticity for new tasks
- Memory usage
- May not fit different architectures
- Computation burden
- Memory usage
- Computation burden

Agenda

- Background on Graph Representation Learning
- Motivation of Continual Graph Learning
- Problem setup of CL and CGL
- CL techniques & CGL techniques
- Evaluation Metrics & CGL benchmarks
- Future directions

Evaluation metrics

Performance Matrix: model's performance (accuracy, f1 score, etc.) after learning each task, the most thorough metric

$$M^p \in R^{T \times T}$$

T: number of tasks in the sequence

$M_{i,j}^p$: Performance on task j after learning from task 1 to i

Evaluation metrics

Performance Matrix: model's performance (accuracy, f1 score, etc.) after learning each task, the most thorough metric

$$M^p \in R^{T \times T}$$

T: number of tasks in the sequence

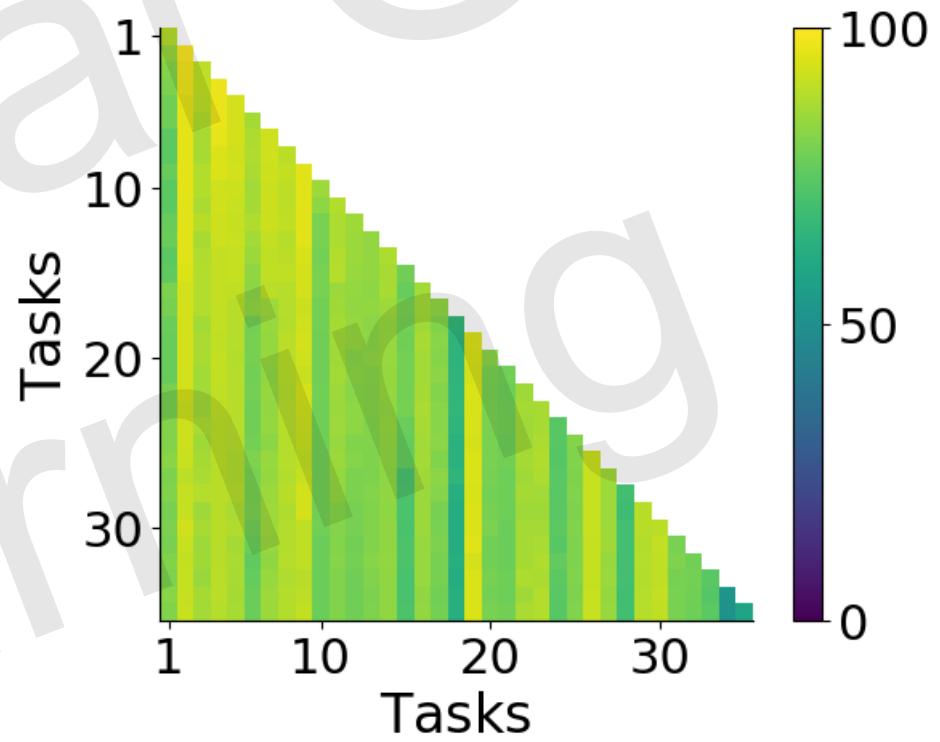
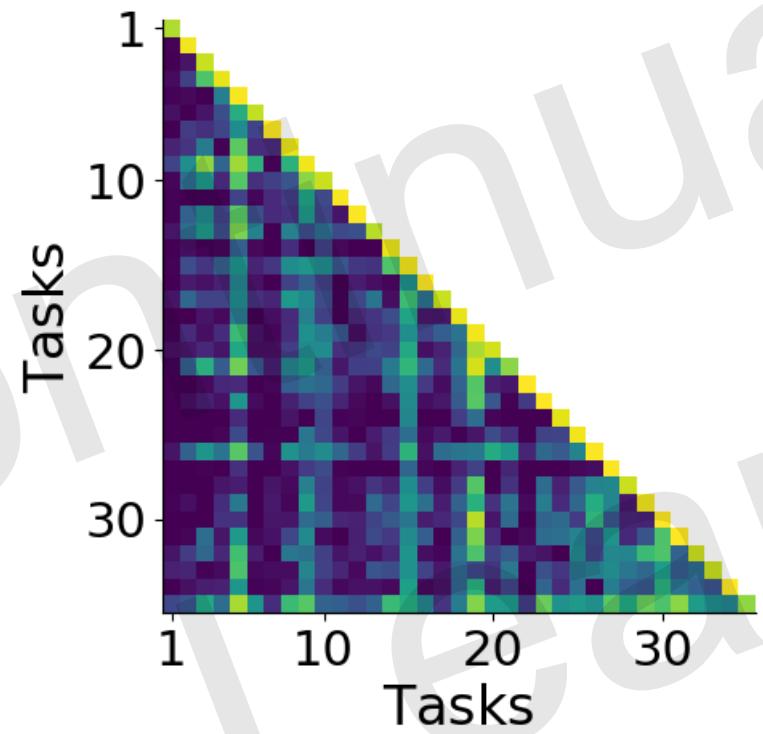
$M_{i,j}^p$: Performance on task j after learning from task 1 to i

An example

	T1	T2	T3
T1	100%		
T2	90%	98%	
T3	60%	70%	99%

Evaluation metrics

Can be visualized



Evaluation metrics

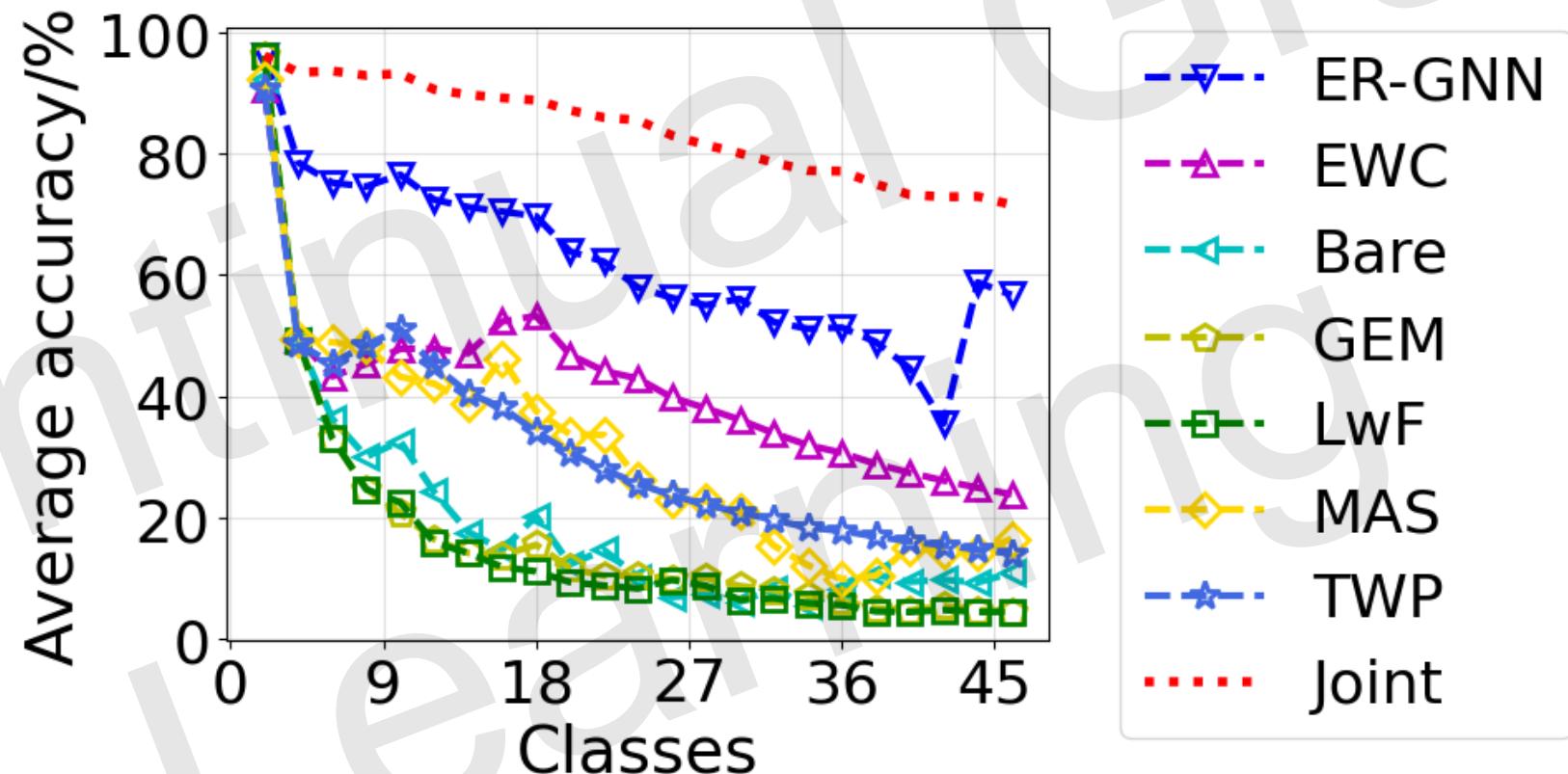
Average Performance (AP): Performance (e.g. accuracy, f1 score) averaged over currently learnt tasks, applicable after learning each task:

$$\left\{ \frac{\sum_{j=1}^i M_{i,j}^p}{i} \mid i = 1, \dots, T \right\}$$

	T1	T2	T3
T1	100%		
T2	90%	98%	
T3	60%	70%	99%

$$\frac{(60\% + 70\% + 99\%)}{3} = 76.3\%$$

Learning dynamics (curve of AP)



Evaluation metrics

Average Forgetting (AF): Performance decrease averaged over all learnt tasks, applicable from the second task

Forgetting on task i

$$\left\{ \frac{\sum_{j=1}^{i-1} M_{i,j}^p - M_{j,j}^p}{i-1} \mid i = 2, \dots, T \right\}$$

	T1	T2	T3
T1	100%		
T2	90%	98%	
T3	60%	70%	99%

$$\frac{(60\% - 100\%) + (70\% - 98\%)}{3-1} = -29\%$$

Average Forgetting (AF) another definition: Performance decrease averaged over all learnt tasks, applicable from the second task

$$f_j = \max_{k=1, \dots, i-1} M_{k,j}^p - M_{i,j}^p$$

AF after each task

$$\left\{ \frac{\sum_{j=1}^{i-1} f_j}{i-1} \mid i = 2, \dots, T \right\}$$

	T1	T2	T3
T1	75%		
T2	90%	98%	
T3	60%	70%	99%

Evaluation metrics

$$\frac{(60\% - 90\%) + (70\% - 98\%)}{3-1} = -24\%$$

Forward transfer

Benefit from learning on all
the (j-1) tasks before j

performance on
task j with random
model initialization

$$\frac{\sum_{j=2}^T M_{j-1,j}^p - \bar{b}_j}{T-1}$$

CGLB: Benchmark Tasks for Continual Graph Learning

Benchmark tasks



N-CGL

G-CGL



N-CGL

Datasets for Task-IL & Class-IL

CoraFull-CL: Classification on new classes of articles

Arxiv-CL: Classification on new classes of articles

Reddit-CL: Classification on new classes of communities

Products-CL: Classification on new classes of products

G-CGL

N-CGL

Datasets for Task-IL & Class-IL

CoraFull-CL: Classification on new classes of articles

Arxiv-CL: Classification on new classes of articles

Reddit-CL: Classification on new classes of communities

Products-CL: Classification on new classes of products

G-CGL

Datasets for Task-IL

Datasets for Class-IL

N-CGL

Datasets for Task-IL & Class-IL

CoraFull-CL: Classification on new classes of articles

Arxiv-CL: Classification on new classes of articles

Reddit-CL: Classification on new classes of communities

Products-CL: Classification on new classes of products

G-CGL

Datasets for Task-IL

SIDER-tIL: Prediction on new molecule properties

Tox21-tIL: Prediction on new molecule properties

Aromaticity-CL: Classification on new classes of molecules

Datasets for Class-IL

Aromaticity-CL: Classification on new classes of molecules

Benchmark tasks



Table 1: The detailed statistics of the constructed benchmark datasets for N-CGL.

Benchmark datasets	CoraFull-CL	Arxiv-CL	Reddit-CL	Products-CL
Data source	CoraFull [34]	OGB-Arxiv ¹	Reddit [17]	OGB-Products ²
Learning scenario	task-IL & class-IL	task-IL & class-IL	task-IL & class-IL	task-IL & class-IL
# nodes	19,793	169,343	227,853	2,449,028
# edges	130,622	1,166,243	114,615,892	61,859,036
# classes	70	40	40	46
# tasks	35	20	20	23
average # nodes per task	660	8,467	11,393	122,451
average # edges per task	4,354	58,312	5,730,794	2,689,523

Benchmark tasks



Table 2: The detailed statistics of the constructed benchmark datasets for G-CGL.

Benchmark datasets	SIDER-tIL	Aromaticity-CL	Tox21-tIL
Data source	SIDER [53]	PubChemBioAssayAromaticity [54]	Tox21 ³
Learning scenario	task-IL	task-IL & class-IL	task-IL
# graphs	1,427	3,868	7,831
# nodes	48,006	115,061	145,459
# edges	100,912	253,018	302,190
# classes	27	30	12
# tasks	27	15	12
average # graphs per task	53	155	653
average # nodes per task	1,778	7,671	12,122
average # edges per task	3,737	16,868	25,183

Benchmark results

1. Task-IL & Class-IL
2. Node-level & Graph-level
3. W/WO Inter-task edges
4. Comparisons among different baselines
5. Visualization results

Toolkit

1. Continual learning pipelines
2. Implementations of different baselines
3. Visualization tools

<https://github.com/QueuQ/CGLB>

A Graph-level Benchmark

Catastrophic Forgetting in Deep Graph Networks: an Introductory Benchmark for Graph Classification

	MNIST	CIFAR10	OGBG-PPA
Size	70000	60000	158100
Node Attrs.	3	5	0
Edge Attrs.	0	0	7
Classes	10	10	37
Avg $ \mathcal{V}_g $	70,57	117,63	243,4
Avg $ \mathcal{E}_g $	564,63	941,07	2266,1
Data Split	55K/5K/15K	45K/5K/15K	49%/29%/22%
Class Split	2+2+2+2+2	2+2+2+2+2	17+5+5+5+5

[diningphil/continual learning for graphs \(github.com\)](https://github.com/diningphil/continual-learning-for-graphs)

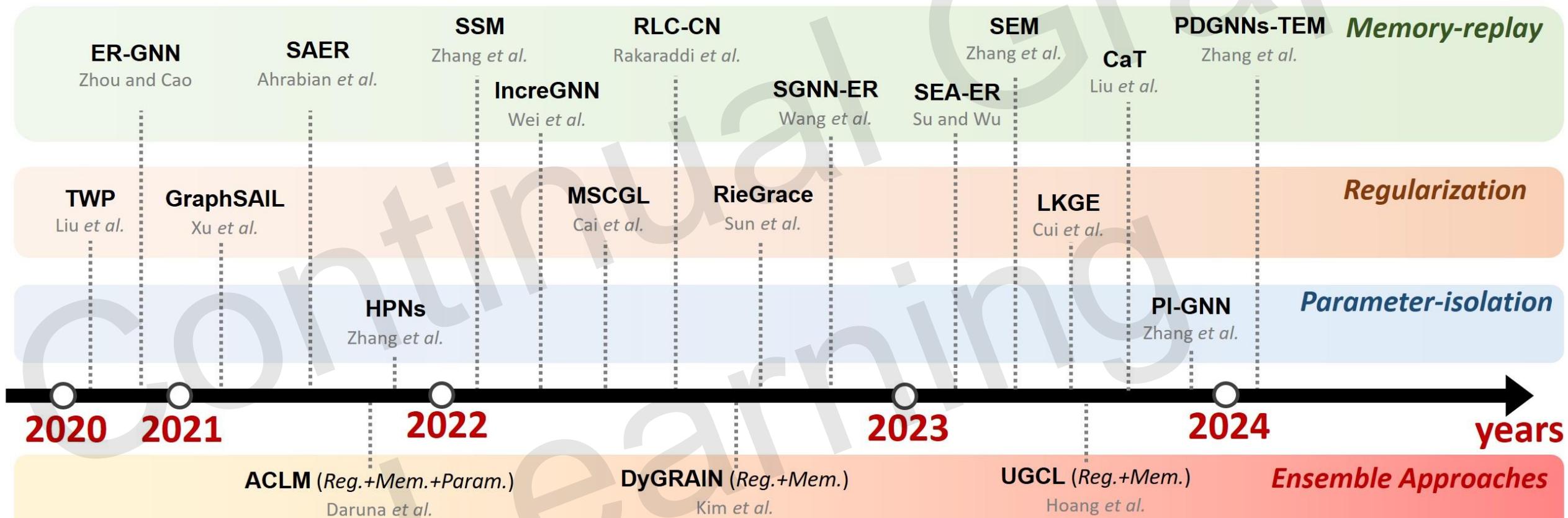
Our Survey

Continual Learning on Graphs: Challenges, Solutions, and Opportunities

Method	Applications	Task Granularity	Technique	Characteristics
TWP [6]	General	Node/Graph	Reg.	Preserve the topology learnt from previous tasks
RieGrace [71]	General	Node	Reg.	Maintain previous knowledge via knowledge distillation
GraphSAIL [72]	Recommender Systems	Node	Reg.	Local and global structure preservation, node information preservation
MSCGL [73]	General	Node	Reg.	Parameter changes orthogonal to previous parameters
LKGE [74]	Knowledge Graph	Node	Reg.	Alleviating forgetting issue with l2 regularization
ER-GNN [75]	General	Node	Mem.	Replay representative nodes
SSM [76]	General	Node	Mem.	Replay representative sparsified computation subgraphs
SEM [77]	General	Node	Mem.	Sparsify computation subgraphs based on information bottleneck
PDGNNs-TEM [78]	General	Node	Mem.	Replay representative topology-aware embeddings
IncreGNN [79]	General	Node	Mem.	Replay nodes according to their influence
RLC-CN [80]	General	Node	Mem.	Model structure adaption and dark experience replay
SGNN-ER [81]	General	Node	Mem.	Model retraining with generated fake historical data
SAER [82]	Recommender System	Node	Mem.	Buffer the representative user-item pairs based on reservoir sampling
SEA-ER [83]	General	Node	Mem.	Minimize the structural difference between the memory buffer and the original graph
CaT [84]	General	Node	Mem.	Train the model solely on balanced condensed graphs from all tasks
HPNs [3]	General	Node	Para.	Extracting and storing basic features to encourage knowledge sharing across tasks, model expanding to accommodate new patterns
PI-GNN [85]	General	Node	Para.	Separate parameters for encoding stable and changed graph parts
DyGRAIN [86]	General	Node	Mem.+Reg.	Alleviate catastrophic forgetting and concept shift of previous task nodes via memory replay and knowledge distillation
ACLM [87]	Knowledge Graph	Node	Mem.+Reg.+Para.	Adapting general CL techniques to CGL tasks
UGCL [88]	General	Node/Graph	Mem.+Reg.	Memory replay & local/global structure preservation

Our Survey

Continual Learning on Graphs: Challenges, Solutions, and Opportunities



Agenda

- Background on Graph Representation Learning
- Motivation of Continual Graph Learning
- Problem setup of CL and CGL
- CL techniques & CGL techniques
- Evaluation Metrics & CGL benchmarks
- Future directions

Remaining challenges & Future directions

- Trade-off between Effectiveness and Space Complexity
- Dependency on the Task Boundaries
- Extension to other Modalities (Heterogeneous graphs)
- Extension to more graph related tasks
- Into large models, on large graphs

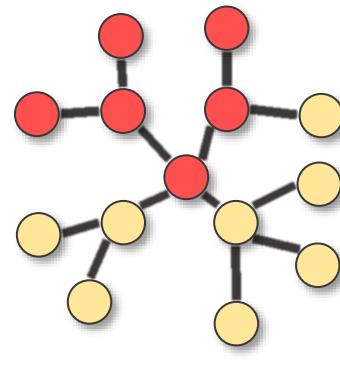
Remaining challenges & Future directions

Trade-off between Effectiveness and Space/Computation Complexity

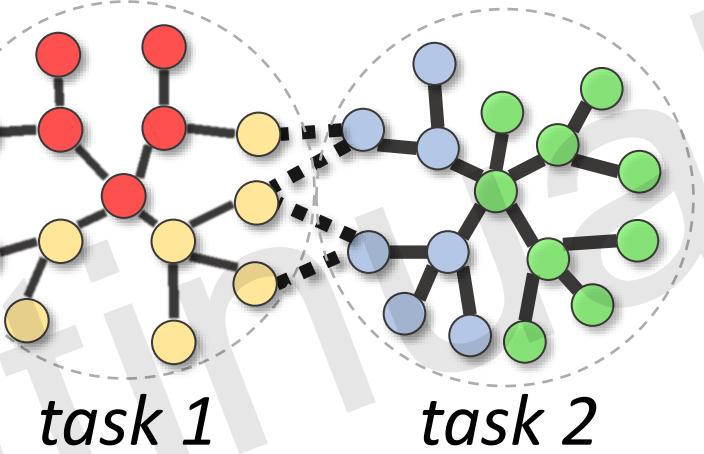
- **Regularization:** No extra memory buffer. Poor plasticity for new tasks.
- **Parameter-isolation:** Extra memory consumption for network expansion. Good performance.
- **Memory-replay:** Extra memory consumption for representative data. Good performance.

Remaining challenges & Future directions

Dependency on the Task Boundaries



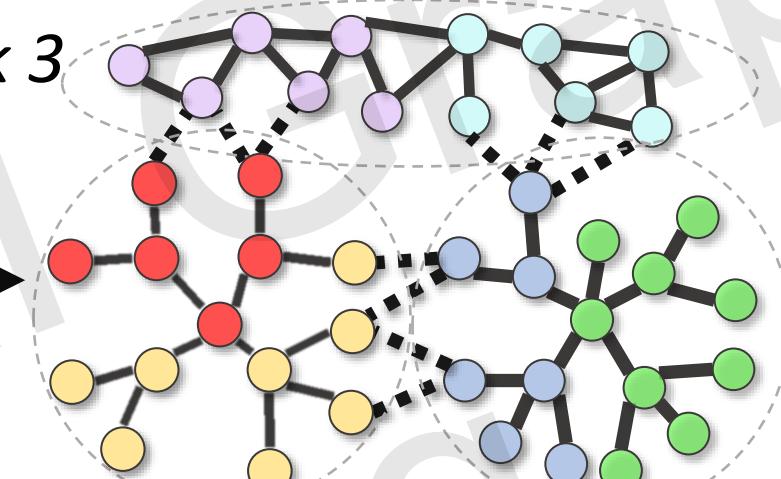
task 1



task 1

task 2

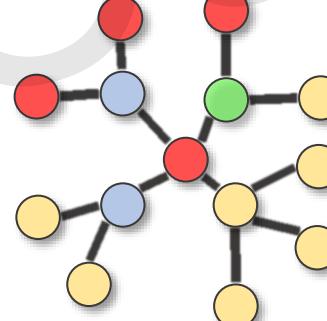
task 3



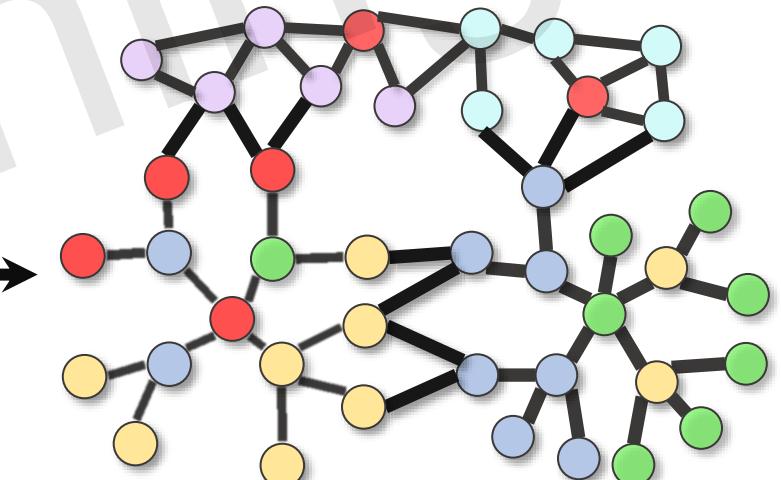
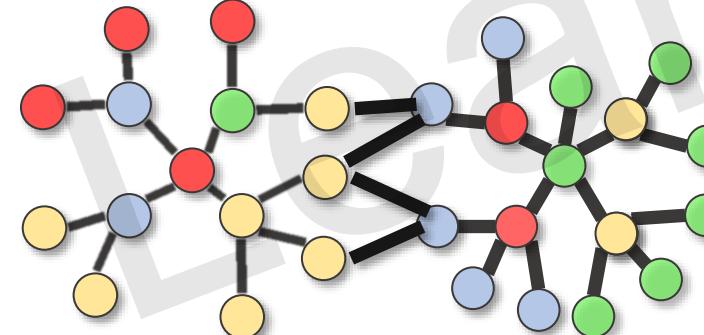
task 1

task 2

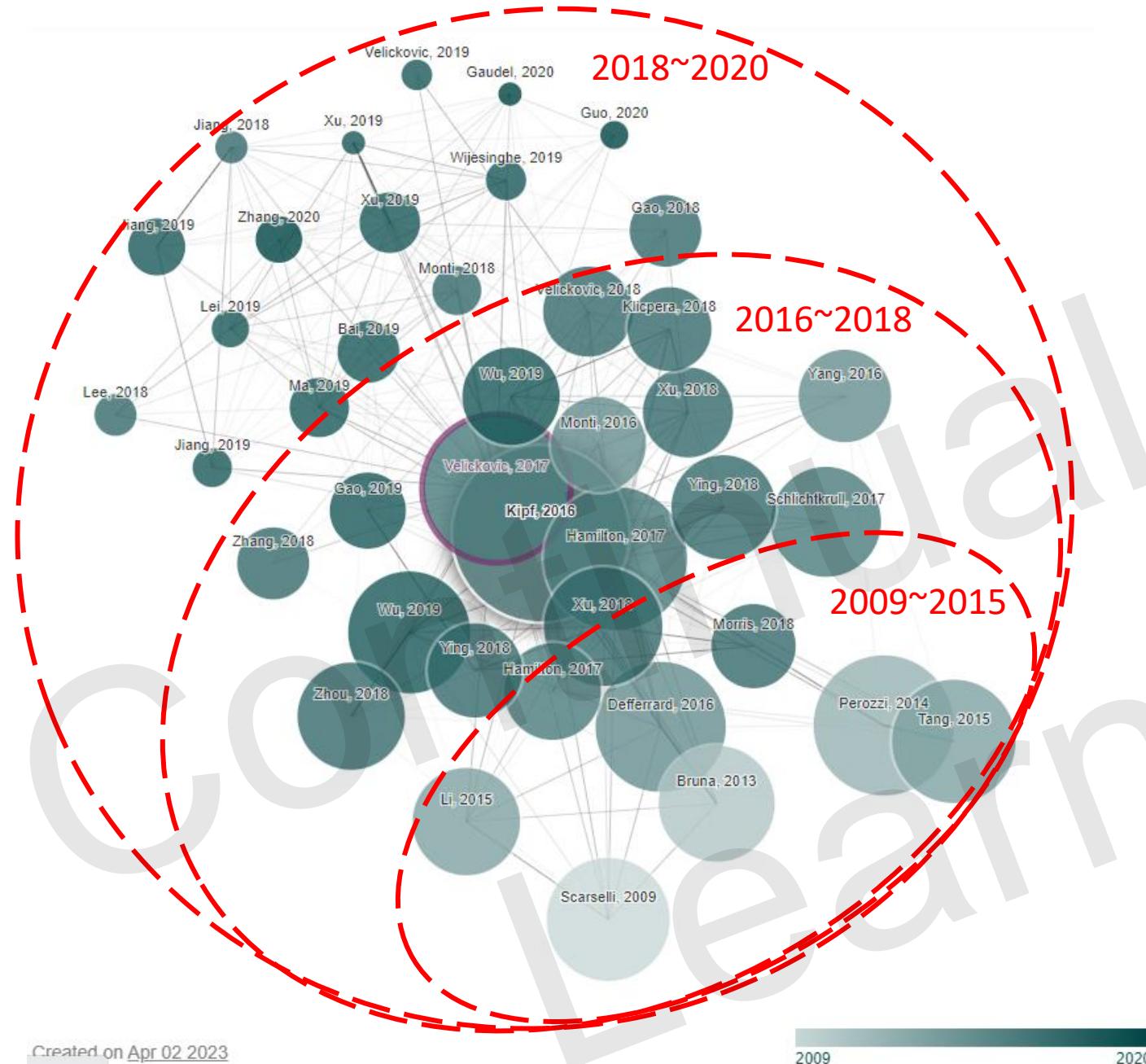
Explicit Task
Boundaries



Future directions



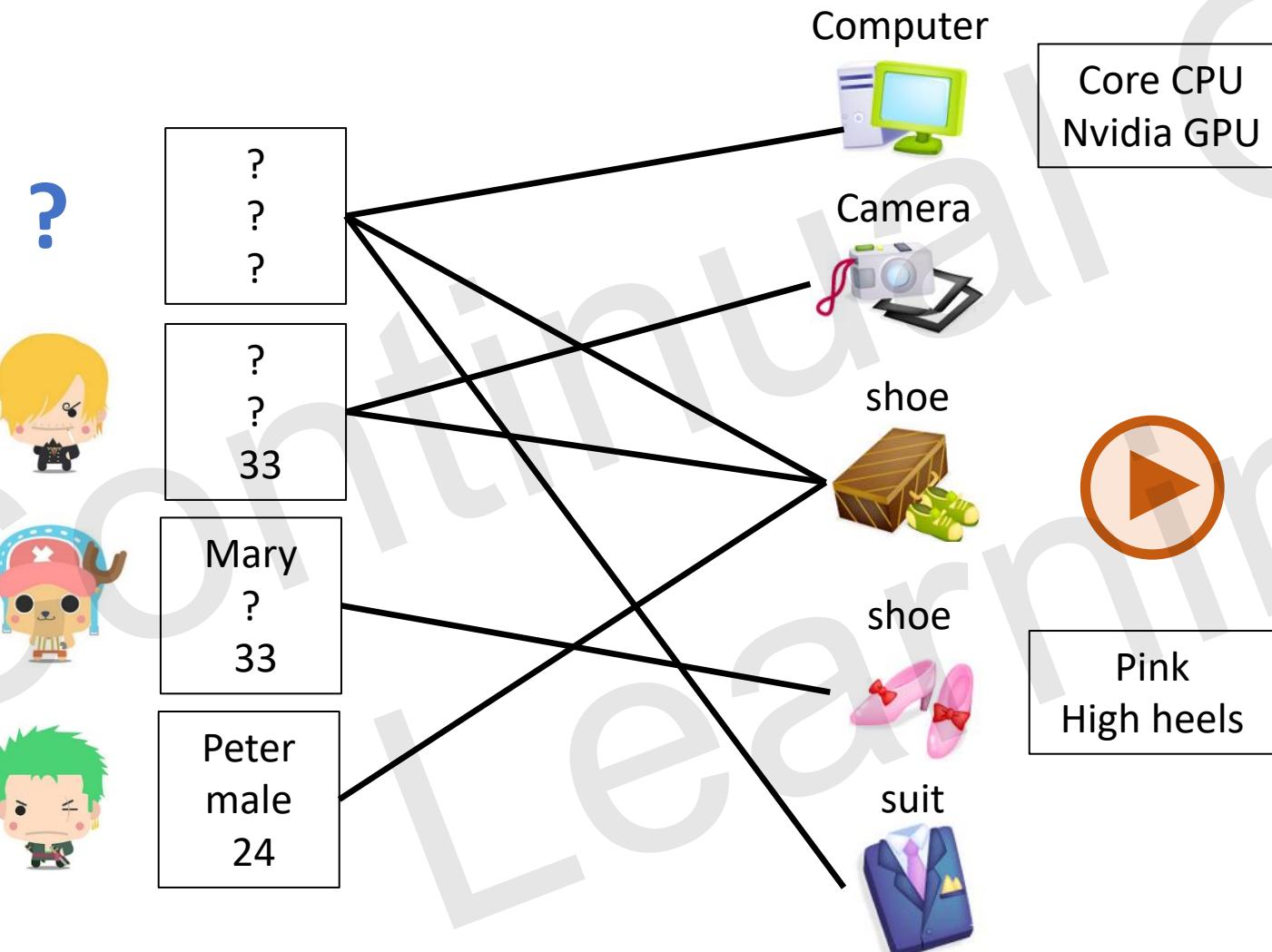
Implicit Task
Boundaries



- Research may focus on different topics in different years.
 - Papers on new and old topics are blended without clear boundaries.
 - When to treat the new data as new tasks?
 - Techniques from other areas may help!

Remaining challenges & Future directions

Extension to other Modalities (Heterogeneous Graphs)

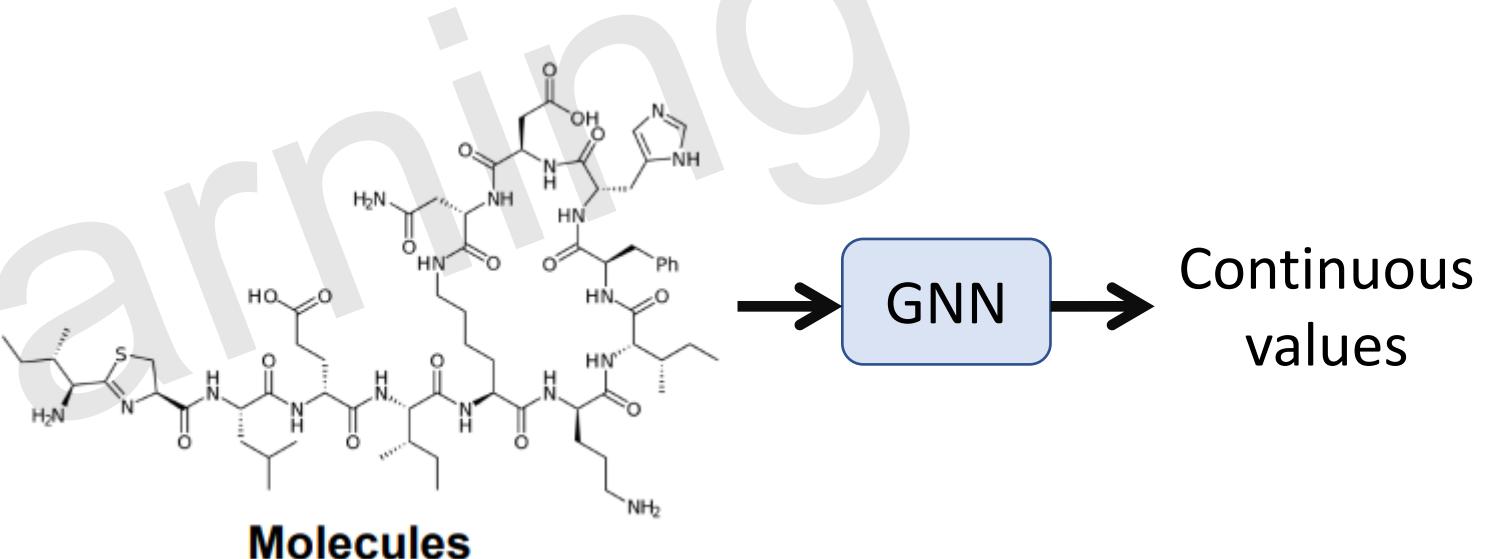
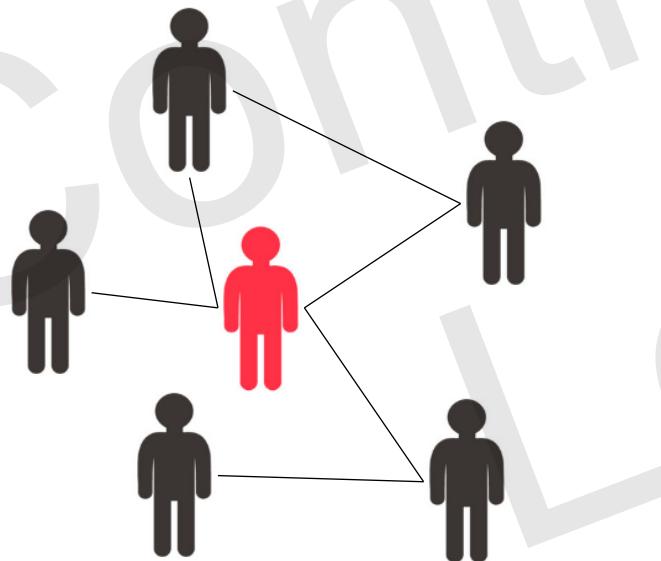


- Different Modalities
- Different number of modalities
- **Can learning on one modality benefit the others?**

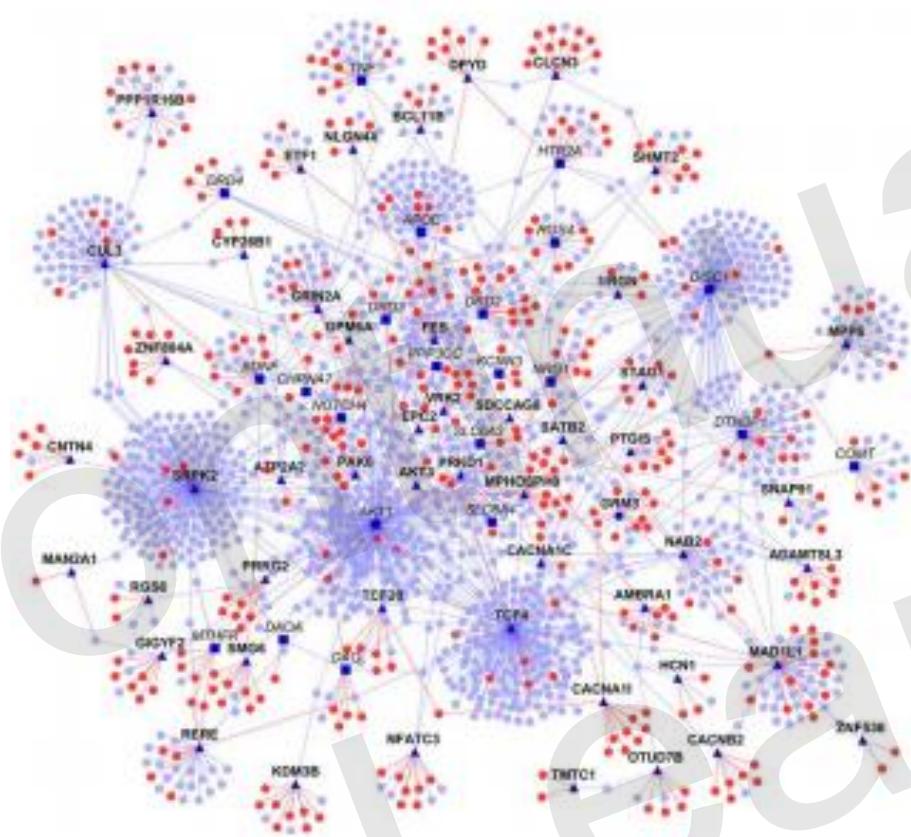
Remaining challenges & Future directions

Extension to more graph related tasks

- Anomaly detection
- Graph/Node regression
- Etc.



Graph data are pervasive in our world Into large models, on large graphs



- Graphs could grow to be very large gradually
 - Models can also grow accordingly

<https://www.genengnews.com/insights/protein-protein-interactions-get-a-new-groove-on/>

Further readings

Gilmer, Justin, et al. "Neural message passing for quantum chemistry." International conference on machine learning. PMLR, 2017.
<https://proceedings.mlr.press/v70/gilmer17a>

For understanding the basic logic of GNNs

Parisi, German I., et al. "Continual lifelong learning with neural networks: A review." *Neural networks* 113 (2019): 54-71.

<https://www.sciencedirect.com/science/article/pii/S0893608019300231>

For an overview of continual learning

Van de Ven, Gido M., and Andreas S. Tolias. "Three scenarios for continual learning." arXiv preprint arXiv:1904.07734 (2019).

<https://arxiv.org/abs/1904.07734>

For an understanding of different incremental scenarios

Zhang, Xikun, Dongjin Song, and Dacheng Tao. "CGLB: Benchmark tasks for continual graph learning." Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track. 2022.

<https://openreview.net/forum?id=5wNiIDynDF>

For an overview of CGL settings and challenges