
Appendix: Hierarchical Prototype Networks for Continual Graph Representation Learning

Anonymous Author(s)

Affiliation

Address

email

1 In this Appendix, we provide the details of theoretical analysis in Section 1, implementation details
2 in Section 2 and additional experimental results and analysis in Section 3. Besides, we also a
3 thorough discussion about related works in Section 4, which is not included in the paper due to space
4 limitations.

5 1 Details of Theoretical Analysis

6 1.1 Overview

7 The main theoretical results are briefly introduced in the paper. In this section, we provide detailed
8 explanations and proofs for the theoretical results. Specifically, we will first provide proofs and de-
9 tailed analysis on the upper bound of the number of different prototypes, as well as the corresponding
10 memory consumption upper bound. Then we will provide proofs and analysis on the task distance
11 preserving and continual learning capability.

12 At below, Definition 1, Lemma 2, Lemma 3, Lemma 4, and Corollary 1 are from existing knowledge
13 ranging from geometry to linear algebra. The other parts are of our own contributions.

14 1.2 Memory consumption upper bound

15 Due to the mechanism to create new prototypes for newly emerging knowledge extracted from the
16 data, the memory consumption will gradually increase. However, because of the normalization
17 applied to the prototypes, the prototype space is constrained, and there exists an upper bound for the
18 memory consumption. This can be intuitively understood as the number of points with distance larger
19 than a threshold is limited on a n -dimensional hypersphere. To formally formulate this, we will first
20 give several definitions.

21 **Definition 1** (Spherical code). *A spherical code $S(n, N, t)$, with parameters (n, N, t) is defined as*
22 *the set of N points on the unit hypersphere in an n -dimension space for which the dot product of unit*
23 *vectors from the origin to any two points is larger than or equal to t .*

24 In our model, the prototypes of different levels can be viewed as spherical codes in their own hidden
25 space, as they are normalized into unit vectors. Specifically, taking the atom prototypes as an example,
26 given the dimension d_a and the threshold t_A , the set of atom prototypes \mathbb{P}_A can be denoted as a
27 spherical code $\mathbb{P}_A = S_A(d_a, N_A, 1 - t_A)$, where N_A is the cardinality of \mathbb{P}_A . Then the upper bound
28 of the number of atom prototypes given d_a and t_A is equal to the maximal cardinality of $S(n, N, t)$
29 given n and t . As the area of the n -dimensional unit sphere surface is limited, it is obvious that
30 there exists a maximal N given a certain n and t , denoted as $\max_N S(d_a, N, 1 - t_A)$. However,
31 finding $\max_N S(d_a, N, 1 - t_A)$ is a complex sphere packing problem, and there is not yet a general
32 formulation of the maximal N for an arbitrary n . Therefore, given the number of two different AFEs
33 as l_a and l_r , we can formulate the upper bound for the numbers of different prototypes as:

Theorem 1 (Upper bounds for numbers of prototypes).

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

$$n_N \leq \max_N S(d_n, N, 1 - t_N) \quad \text{and} \quad n_C \leq \max_N S(d_c, N, 1 - t_C) \quad (2)$$

Although the general formulation is not available for an arbitrary dimension, we can specially compute $\max_N S(d_a, N, 1 - t_A)$ for certain n s, and verify it with experiments. For example, when $n = 2$, the distribution becomes distributing points on a circle with unit radius. Then, $\max_N S(d_a, N, 1 - t_A)$ can be obtained by evenly distributing the points on the circle with an interval of t_A . Finally, the explicit value of $\max_N S(d_a, N, 1 - t_A)$ can be formulated as:

$$\max_N S(d_n, N, 1 - t_N) = \frac{2\pi}{\arccos(1 - t_A)}, \quad (3)$$

then we have:

$$n_A \leq (l_a + l_r) \frac{2\pi}{\arccos(1 - t_A)}. \quad (4)$$

And the upper bound of the number of N- and C-prototypes can be formulated similarly. The above results are used in Section 3.7 in the paper.

1.3 Task distance preserving

In continual learning, the key challenge is to overcome the catastrophic forgetting, which refers to the performance degradation on previous tasks after training the model on new tasks. Based on our model design, we formulate this as: whether learning new tasks affect the representations the model generates for old task data. First, we give definitions on the tasks and task distances:

Definition 2 (Task set). *The p -th task in a sequence is denoted as \mathcal{T}^p and contains a subgraph \mathcal{G}_p consisting of nodes belonging to some new categories. We denote the associated node set and adjacency matrix as \mathbb{V}_p and A_p . Each $v_p^i \in \mathbb{V}_p$ has a feature vector $\mathbf{x}(v_p^i)$ and a label $y(v_p^i)$.*

Then, the reason for catastrophic forgetting is that different tasks in a sequence are drawn from heterogeneous distributions, making the model sequentially trained on different tasks unable to maintain satisfying performances on previous tasks. Therefore, given the definition of the tasks (Definition 2), we then give a formal definition to quantify the difference between two tasks.

Definition 3 (Task distance). *We define the distance between two tasks as the set distance between the node sets of these two tasks, i.e.*

$$\text{dist}(\mathbb{V}_p, \mathbb{V}_q) = \inf \|\mathbf{x}(v_p^i) - \mathbf{x}(v_q^j)\|, \forall v_p^i \in \mathbb{V}_p, v_q^j \in \mathbb{V}_q.$$

Lemma 1. *The distance between any two tasks is non-negative, i.e. $\forall i, j \in \{1, \dots, M^T\}, \text{dist}(\mathbb{V}_p, \mathbb{V}_q) \geq 0$, where M^T is the number of tasks contained in the sequence.*

The real-world data could be complex and sometimes may even contain noises that are impossible for any model to learn, which needs extra considerations when justifying the effectiveness of the model. Formally, we give the definition of the contradictory data.

Definition 4 (Contradictory data). *$\forall v_p^i \in \mathbb{V}_p, p = 1, \dots, M^T$, if $\exists v_q^j \in \mathbb{V}_q, j = 1, \dots, M^T$, st. $\forall l \in \mathbb{N}^*, \forall u \in \mathcal{N}^l(v_p^i)$ and $\forall v \in \mathcal{N}^l(v_q^j), \mathbf{x}(u) = \mathbf{x}(v)$ but $y(v_p^i) \neq y(v_q^j)$, then we say $(v_p^i, y(v_p^i))$ and $(v_q^j, y(v_q^j))$ are contradictory data, as it is contradictory for any model to give different predictions for one node based on the same node features and graph structures. (\mathbb{N}^* denotes the set of non-negative integers)*

Remark 1. *Contradictory data is ignored or simply regarded as outliers in previous works, but in this work, we explicitly analyze its affect for the comprehensiveness of our theory. contradictory data has different situations. If v_p^i and v_q^j are from different tasks, then $y(v_p^i) \neq y(v_q^j)$ is plausible. Because they may be describing a same thing from different aspects. For example, an article from the citation network may be both categorized as 'physics related' and 'computer science related'. In this*

73 situation, it would be easy to add an task indicator to the feature of the node, then the feature of v_p^i
 74 and v_q^j are no longer equal and are not contradictory data anymore.

75 But within one task, contradictory data are most likely to be wrongly labeled, e.g. it does not make
 76 sense if an article is both 'related to physics' and 'not related to physics'.

77 Besides the distance between tasks, the distance between the embeddings obtained by the AFEs will
 78 also be a crucial concept in the proof.

79 **Definition 5** (Embedding distance). Each input node v_p^i is given a set of atomic embeddings
 80 $\mathbb{E}_A(v_p^i) = \mathbb{E}_A^{\text{node}}(v_p^i) \cup \mathbb{E}_A^{\text{struct}}(v_p^i)$, where $\mathbb{E}_A^{\text{node}}(v_p^i) = \{\mathbf{a}_i^j | j \in \{1, \dots, l_a\}\}_p$ containing the atomic
 81 node embeddings of v_p^i and $\mathbb{E}_A^{\text{struct}}(v_p^i) = \{\mathbf{r}_i^j | k \in \{1, \dots, l_r\}\}_p$ containing the atomic structure
 82 embeddings. $\mathbf{a}_i^j \in \mathbb{R}^{d_a}$ and $\mathbf{r}_i^j \in \mathbb{R}^{d_r}$. To define the distance between representations of two nodes,
 83 we concatenate the atomic embeddings of each node into a single vector in a higher dimensional
 84 space, i.e. each node v_p^i corresponds to a latent vector $\mathbf{z}_p^i = [\mathbf{a}_i^1; \dots; \mathbf{a}_i^{l_a}; \mathbf{r}_i^1; \dots; \mathbf{r}_i^{l_r}] \in \mathbb{R}^{l_a \times d_a + l_r \times d_r}$.
 85 Then we define the distance between representations of two nodes v_p^i and v_q^j as the Euclidean distance
 86 between their corresponding latent vector \mathbf{z}_p^i and \mathbf{z}_q^j , i.e. $\text{dist}(\mathbf{z}_p^i, \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2$.

87 Then we will give some explanations on the linear algebra related theories.

88 **Lemma 2** (Bounds for real quadratic forms). Given a real symmetric matrix \mathbf{A} , and an arbitrary
 89 real vector variable \mathbf{x} , we can give

$$90 \lambda_{\min} \leq \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \leq \lambda_{\max},$$

91 where λ_{\min} and λ_{\max} are the minimum and maximum eigenvalues of matrix \mathbf{A} .

92 **Lemma 3** (Real symmetric matrix). For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{A}^T \mathbf{A} \in \mathbb{R}^{n \times n}$ is a real symmetric
 93 matrix, $\text{rank}(\mathbf{A}^T \mathbf{A}) = \text{rank}(\mathbf{A})$, and the non-zero eigenvalues of $\mathbf{A}^T \mathbf{A}$ are squares of the non-zero
 94 singular values of \mathbf{A} .

95 **Lemma 4** (Rank and number of non-zero singular values). For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, the number of
 96 non-zero singular values equals the rank of \mathbf{A} , i.e. $\text{rank}(\mathbf{A})$

97 **Corollary 1.** For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. Without loss of generality, we assume $n \leq m$. If \mathbf{A} is column
 98 full rank, i.e. $\text{rank}(\mathbf{A}) = n$, then \mathbf{A} has n non-zero singular values. Besides, $\text{rank}(\mathbf{A}^T \mathbf{A}) = n$,
 99 and $\mathbf{A}^T \mathbf{A}$ has n non-zero singular values.

100 Given the explanations above, we then derive the bound for the change of the distance among data,
 101 which will be further used for analyzing the separation of data from different tasks.

102 **Lemma 5** (Embedding distance bound). Given two nodes $v_p^i \in \mathbb{V}_p$ and $v_q^j \in \mathbb{V}_q$ with vertex
 103 feature $\mathbf{x}(v_p^i), \mathbf{x}(v_q^j) \in \mathbb{R}^{d_v}$, their multi-hop neighboring node sets are denoted as $\bigcup_{l \in \mathbb{N}^*} \mathcal{N}^l(v_p^i)$ and
 104 $\bigcup_{l \in \mathbb{N}^*} \mathcal{N}^l(v_q^j)$. The AFEs for generating atomic embeddings are $\text{AFE}_{\text{node}} = \{\mathbf{A}_i \in \mathbb{R}^{d_a \times d_v} | i \in$
 105 $\{1, \dots, l_a\}$ and $\text{AFE}_{\text{struct}} = \{\mathbf{R}_j \in \mathbb{R}^{d_r \times d_v} | j \in \{1, \dots, l_r\}$, corresponding to matrices for atomic
 106 node embeddings and atomic structure embeddings, respectively. Then, the square distance

$$107 \text{dist}^2(\mathbf{z}_p^i - \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2^2 \geq \lambda_{\min}(\|\mathbf{x}(v_p^i) - \mathbf{x}(v_q^j)\|_2^2 + \sum_{k=1}^{l_r} \|\mathbf{x}(u_k) - \mathbf{x}(v_k)\|_2^2),$$

108 if $l_a \times d_a + l_r \times d_r \geq d_v$, where u_k are nodes sampled from $\bigcup_{l \in \mathbb{N}^*} \mathcal{N}^l(v_p^i)$, v_k are nodes sampled from

109 $\bigcup_{l \in \mathbb{N}^*} \mathcal{N}^l(v_q^j)$, λ_i are the eigenvalues of $\mathbf{W}^T \mathbf{W}$, and $\mathbf{W} \in \mathbb{R}^{(l_r+1)d_v \times (l_a d_a + l_r d_r)}$ is constructed with
 110 the matrices in AFE_{node} and $\text{AFE}_{\text{struct}}$. Specifically, \mathbf{W} is a block matrix constructed as follows:

- 111 1. $\mathbf{W}_{1:l_a d_a, 1:d_v}$ are filled by the concatenation of $\{\mathbf{A}_i | i = 1, \dots, l_a\}$, i.e. $[\mathbf{A}_1; \dots; \mathbf{A}_{l_a}] \in \mathbb{R}^{l_a d_a \times d_v}$.
- 112 2. For $\mathbf{W}_{l_a d_a + 1:l_a d_a + l_r d_r, 1:(l_r+1)d_v}$, the construction is first filling
 113 $\mathbf{W}_{l_a d_a + (k-1)d_r : l_a d_a + k d_r, k d_v : (k+1)d_v}$ with \mathbf{R}_k , $k = 1, \dots, l_r$.
- 114 3. For other parts, fill with zeros.

115 *Proof.* Given vertex v_p^i , we concatenate its feature vector with the l_r neighbors sampled from
 116 $\bigcup_{l \in \mathbb{N}^*} \mathcal{N}^l(v_p^i)$, i.e. $\mathbf{x}'_{p,i} = [\mathbf{x}(v_p^i); \mathbf{x}(u_1); \dots; \mathbf{x}(u_{l_r})] \in \mathbb{R}^{(l_r+1)d \times 1}$, $u_j \in \bigcup_{l \in \mathbb{N}^*} \mathcal{N}^l(v_p^i)$. Then with the
 117 constructed block matrix \mathbf{W} , we could formulate the generation of \mathbf{z}_p^i as:

$$118 \quad \mathbf{z}_p^i = \mathbf{W} \mathbf{x}'_{p,i}.$$

119 Similarly, we can formulate \mathbf{z}_q^j for another vertex v_q^j .

120 And their distance can be formulated as:

$$121 \quad \text{dist}(\mathbf{z}_p^i, \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2 = \sqrt{(\mathbf{z}_p^i - \mathbf{z}_q^j)^T (\mathbf{z}_p^i - \mathbf{z}_q^j)}$$

$$\begin{aligned} 122 & (\mathbf{z}_p^i - \mathbf{z}_q^j)^T (\mathbf{z}_p^i - \mathbf{z}_q^j) \\ 123 &= (\mathbf{W} \mathbf{x}'_{p,i} - \mathbf{W} \mathbf{x}'_{q,j})^T (\mathbf{W} \mathbf{x}'_{p,i} - \mathbf{W} \mathbf{x}'_{q,j}) \\ 124 &= (\mathbf{W} (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j}))^T (\mathbf{W} (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})) \\ 125 &= (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})^T \mathbf{W}^T \mathbf{W} (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j}) \end{aligned}$$

126

127 According to lemma 3, $\mathbf{W}^T \mathbf{W}$ is a real symmetric matrix, with lemma 2, we have

$$\begin{aligned} 128 & \frac{(\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})^T \mathbf{W}^T \mathbf{W} (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})}{(\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})^T (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})} \\ 129 & \geq \lambda_{\min} \end{aligned}$$

130 According to lemma 3, with $l_a d_a + l_r d_r \geq (l_r + 1) d_v$ and the constraint of column full rank on \mathbf{W} ,
 131 $\mathbf{W}^T \mathbf{W} \in \mathbb{R}^{(l_r+1) \times (l_r+1)}$ has $l_r + 1$ positive eigenvalues, thus $\lambda_{\min} > 0$.

132 Then we decompose $(\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})^T (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})$,

$$\begin{aligned} 133 & (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j})^T (\mathbf{x}'_{p,i} - \mathbf{x}'_{q,j}) = \sum_{k=0}^{(l_r+1)d-1} ((\mathbf{x}'_{p,i})_k - (\mathbf{x}'_{q,j})_k)^2 \\ 134 &= \sum_{k=1}^{(l_r+1)d_v} ((\mathbf{x}'_{p,i})_k - (\mathbf{x}'_{q,j})_k)^2 \\ 135 &= \sum_{k=1}^{d_v} ((\mathbf{x}'_{p,i})_k - (\mathbf{x}'_{q,j})_k)^2 + \sum_{k=1}^{l_r} \sum_{m=k d_v + 1}^{(k+1)d_v} ((\mathbf{x}'_{p,i})_k - (\mathbf{x}'_{q,j})_k)^2 \\ 136 &= \|\mathbf{x}(v_p^i) - \mathbf{x}(v_q^j)\|_2^2 + \sum_{m=1}^{l_r} \|\mathbf{x}(u_m) - \mathbf{x}(\nu_k)\|_2^2 \\ 137 & \therefore \text{dist}^2(\mathbf{z}_p^i - \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2^2 \geq \lambda_{\min} (\|\mathbf{x}(v_p^i) - \mathbf{x}(v_q^j)\|_2^2 + \sum_{k=1}^{l_r} \|\mathbf{x}(u_k) - \mathbf{x}(\nu_k)\|_2^2) \end{aligned}$$

138

□

139 The key point in these theories is that for any task sequence with certain distance among the tasks,
 140 there exists a configuration that ensures HPNs to be capable of preserving the task distance after
 141 projecting the data into the hidden space, so that only the prototypes associated with the current task
 142 are refined and the prototypes corresponding to the other tasks are preserved. Specifically, theorem
 143 on zero-forgetting can be formulated as follows:

144 **Theorem 2** (Task distance preserving). *For HPNs trained on consecutive tasks \mathcal{T}^p and \mathcal{T}^{p+1} .
 145 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 $t_A < \lambda_{\min}(l_r +$
 146 $1) \text{dist}(\mathbb{V}_p, \mathbb{V}_{p+1})$, learning on \mathcal{T}^{p+1} will not modify representations HPNs generate for data from
 147 \mathcal{T}^p , i.e. catastrophic forgetting is avoided.*

148 In Theorem 2, λ_i is eigenvalues of the $\mathbf{W}^T \mathbf{W}$, where \mathbf{W} is the matrix mentioned before constructed
 149 via AFEs. d_v , d_a and d_r are dimensions of data, atomic node embeddings, and atomic structure
 150 embeddings.

151 *Proof.* Following the proofs above, suppose two nodes v_p^i and v_q^j are embedded into \mathbf{z}_p^i and \mathbf{z}_q^j with
 152 the embedding module. Then the distance between \mathbf{z}_p^i and \mathbf{z}_q^j could be formulated as:

$$153 \quad \text{dist}(\mathbf{z}_p^i, \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2 = \sqrt{(\mathbf{z}_p^i - \mathbf{z}_q^j)^T (\mathbf{z}_p^i - \mathbf{z}_q^j)}$$

Dataset	Cornell	Texas	Wisconsin	Cora	Citeseer	Actor	OGB-Arxiv	OGB-Products
# nodes	183	183	251	2,708	3,327	7,600	169,343	2,449,029
# edges	295	309	499	5,429	4,732	33,544	1,166,243	61,859,140
# features	1,703	1,703	1,703	1,433	3,703	931	128	100
# classes	5	5	5	7	6	4	40	47
# tasks	2	2	2	3	3	2	20	23

Table 1: The detailed statistics of 8 datasets used in our experiments.

154 According to lemma 5, we have $\text{dist}^2(\mathbf{z}_p^i, \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2^2 \geq \lambda_{\min}(\|\mathbf{x}(v_p^i) - \mathbf{x}(v_q^j)\|_2^2 +$
155 $\sum_{k=1}^{l_r} \|\mathbf{x}(u_k) - \mathbf{x}(\nu_k)\|_2^2).$

156 $\because v_p^i \in \mathbb{V}_p, v_q^j \in \mathbb{V}_q,$

157 $\therefore \|\mathbf{x}(v_p^i) - \mathbf{x}(v_q^j)\|_2^2 \geq \text{dist}^2(\mathbb{V}_p, \mathbb{V}_q).$

158 Similarly, $\|\mathbf{x}(u_k) - \mathbf{x}(\nu_k)\|_2^2 \geq \text{dist}^2(\mathbb{V}_p, \mathbb{V}_q),$ for $\forall k.$

159 $\therefore \|\mathbf{z}_p^i, \mathbf{z}_q^j\|_2^2 \geq \lambda_{\min}(l_r + 1)\text{dist}^2(\mathbb{V}_p, \mathbb{V}_q)$

160 $\therefore \text{dist}(\mathbf{z}_p^i, \mathbf{z}_q^j) = \|\mathbf{z}_p^i - \mathbf{z}_q^j\|_2 \geq \sqrt{\lambda_{\min}(l_r + 1)\text{dist}^2(\mathbb{V}_p, \mathbb{V}_q)}$

161 \therefore If $t_A < \sqrt{\lambda_{\min}(l_r + 1)\text{dist}^2(\mathbb{V}_p, \mathbb{V}_q)},$ the embeddings of two nodes from two different tasks will
162 not be assigned to same A-prototypes.

163 Above all, if the conditions in Theorem 2 are satisfied, learning on new tasks will not modify the
164 prototypes for previous tasks. Besides, the data from previous tasks will be exactly matched to the
165 correct prototypes after training the model on new tasks. In practice, the conditions may not be easy
166 to be satisfied all the time. However, as mentioned in the paper, the bound given in Theorem 2 is not
167 tight, thus fully satisfying the conditions may not be necessary. Therefore, in the experimental section
168 in the paper, we practically show how the important factors included in these conditions influence
169 the performance (Section 3.6 in the paper). The results demonstrates that the more we satisfy the
170 conditions, the better performance we will obtain, and certain factors (number of AFEs) influence
171 more than the others. \square

172 **Remark 2.** When $\text{dist}(\mathbb{V}_p, \mathbb{V}_q) = 0,$ i.e. there exists a non-empty set $\mathbb{V}_{\cap} = \mathbb{V}_p \cap \mathbb{V}_q,$ st. $\text{dist}(\mathbb{V}_p \setminus$
173 $\mathbb{V}_{\cap}, \mathbb{V}_q \setminus \mathbb{V}_{\cap}) > 0,$ then Theorem 2 holds. As for the \mathbb{V}_{\cap} containing examples exactly same in \mathbb{V}_p
174 and $\mathbb{V}_q,$ there are two situations:

175 1. $\forall v \in \mathbb{V}_{\cap}, y_p(v) = y_q(v),$ where $y_p(\cdot)$ and $y_q(\cdot)$ denote the associated labels in task p and q

176 2. $\exists v \in \mathbb{V}_{\cap}, y_p(v) \neq y_q(v)$

177 For situation 1, \mathbb{V}_{\cap} will not cause the model to forget about the previous task, as these shared data
178 are exactly same and will optimize the model to same direction. For situation 2, if no task indicator
179 is provided, then these data are contradictory data, if task indicator is provided, then the indicator
180 could be merged into the feature vector of the node, i.e. $\mathbf{x}(v_p),$ then v_p will not belong to $\mathbb{V}_{\cap}.$

181 2 Details of Implementation

182 2.1 Datasets and task splitting

183 In this subsection, we introduce the datasets we used and the details of how each dataset is split into
184 different tasks.

185 We use 8 publicly datasets which include 2 citation networks (Cora[29], Citeseer [29], OGB-Arxiv
186 [36, 23]), 3 web page networks (Wisconsin, Cornell, Texas) [25], and 1 actor co-occurrence network
187 (Actor) [25], and one product co-purchasing network (OGB-Products [3]). The detailed statistics of
188 these 8 datasets are summarized in Table 1.

2.1.1 Citation networks

The original Cora [22] and Citeseer [10] are pre-processed by Sen et al. [29] with stemming and removing stop words as well as words with document frequency less than 10. Finally, Cora contains 2708 documents, 5429 links denoting the citations among the documents, and each document is represented with 1433 distinct words. Cora contains 7 classes. For training, 140 documents are selected with 20 examples for each class. The validation set contains 500 documents and the test set contains 1000 examples. In our continual learning setting, the first 6 classes are selected and grouped into 3 tasks (2 classes for each task) in the original order. Citeseer results in 3312 documents with each document being represented with 3703 distinct words, and 4732 links. Citeseer contains 6 classes. 20 documents per class are selected, for training, 500 documents are selected, for validation, and 1000 documents are selected as the test set. For continual learning setting, the documents from 6 classes are grouped into 3 tasks with 2 classes per task in the original order. The Cora and Citeseer datasets can be downloaded via Cora&Citeseer.

The OGB-Arxiv dataset is collected in the Open Graph Benchmark OGB. It is a directed citation network between all Computer Science (CS) arXiv papers indexed by MAG [36]. Totally it contains 169,343 nodes and 1,166,243 edges. Each node is an paper and each directed edge indicates that one paper cites another one. Each paper comes with a 128-dimensional feature vector. The dataset contains 40 classes. As the dataset is not balanced and the numbers of examples in different classes differs significantly, directly grouping the classes into 2-class groups like the Cora and Citeseer will cause certain tasks to be imbalanced. Therefore, we reordered the classes in an descending order according to the number of examples contained in each class, and then group the classes according to the new order. In this way, the number of examples contained in different classes of each task are arranged to be as balanced as possible. Specifically, the class indices of each task are: [[35, 12], [15, 21], [28, 30], [16, 24], [10, 34], [8, 4], [5, 2], [27, 26], [36, 19], [23, 31], [9, 37], [13, 3], [20, 39], [22, 6], [38, 33], [25, 11], [18, 1], [14, 7], [0, 17], [29, 32]].

2.1.2 Web page networks

WebKB dataset is collected from different universities by Carnegie Mellon University. The nodes in the datasets are web pages with bag-of-words representation, and edges are hyperlinks between the pages. The web pages are manually classified into 5 classes including student, project, course, staff, and faculty. Following setting in [25], we use three subsets including Wisconsin with 251 web pages, Cornell with 183 web pages, and Texas with 183 web pages. For all these datasets, 60% nodes are used for training, 20% for validation, and 20% for testing. For each of the web page networks, we constructed 2 tasks with 2 classes per task. The three web page network datasets can be accessed via Web Pages. The balanced splitting of the classes for the three web page networks is [[2, 3], [0, 4]]

2.1.3 Actor co-occurrence network

The actor co-occurrence network is a subgraph of the film-director-actor-writer network [31]. Each node in this dataset corresponds to an author, and the edges between the nodes are co-occurrence on the same Wikipedia pages. The whole dataset contains 7600 nodes and 33544 edges. Each node is accompanied with a feature vector of 931 dimensions. The nodes are classified into 4 classes according to the number of the average monthly traffic of the web page. For this dataset, we also constructed 2 tasks with 2 classes per task. The link to this dataset is Actor. The balanced splitting of the classes is [[0, 1], [2, 3]]

2.1.4 Product co-purchasing network

OGB-Products is also collected in the Open Graph Benchmark OGB, and is an undirected and unweighted graph, representing an Amazon product co-purchasing network link. In total, it contains 2,449,029 nodes and 61,859,140 edges. Nodes represent products sold in Amazon, and edges between two products indicate that the products are purchased together. Node features are generated by extracting bag-of-words features from the product descriptions followed by a Principal Component Analysis to reduce the dimension to 100. 47 top-level categories are used for target labels, in our experiments, we select 46 classes and omit the final class containing only 1 example. Similar to OGB-Arxiv, we reorder the classes in an descending order according to the number of examples contained in each class, and then group the classes according to the new order. The class indices of each tasks are: [[4, 7], [6, 3], [12, 2], [0, 8], [1, 13], [16, 21], [9, 10], [18, 24], [17, 5], [11, 42], [15,

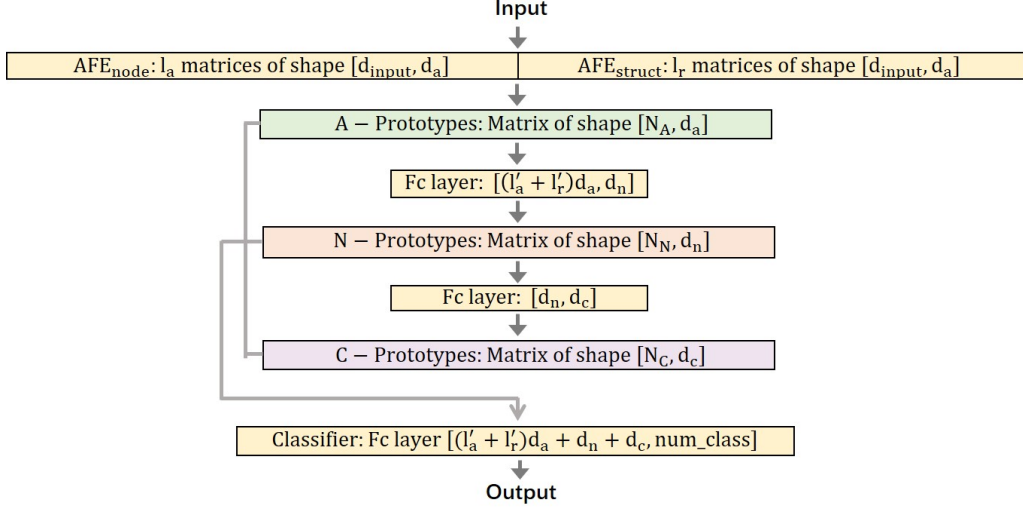


Figure 1: Details of modules in HPNs.

242 20], [19, 23], [14, 25], [28, 29], [43, 22], [36, 44], [26, 37], [32, 31], [30, 27], [34, 38], [41, 35], [39,
243 33], [45, 40]].

2.2 Experiment Setup

245 All models are implemented in PyTorch with SGD optimizer and repeated 5 times on a Nvidia Titan
246 Xp GPU. The average performance and standard deviations are reported for comparison. The network
247 architecture of HPNs is detailed in Figure 1, and the specific values of hyperparameters will be given
248 in the following. The hyperparameters we provide here correspond to the models used in comparisons
249 with the baselines, while in other experiments the hyperparameters are the research objects and will
250 not be kept unchanged.

251 As the sizes of the datasets we used are greatly different, we adopt different hyperparameters for small
252 datasets and large datasets. The small datasets include Cora, Citeseer, Actor, Wisconsin, Cornell, and
253 Texas. The large datasets include OGB-Arxiv and OGB-Products.

254 For the small datasets, we set $l'_a = 1$, $l'_r = 1$, and $h = 2$. We randomly sample 5 one-hop neighbors
255 and 7 two-hop neighbors. The learning rates are managed separately for different modules of the
256 model. For the AFEs, the learning rate is set as 0.1 at the beginning and decays to 0.001 at epoch 35.
257 The learning rate for the prototypes are initialized as 0.1 at epoch 35 and decays to 0.01 at epoch
258 85. And the learning rates for the other trainable parameters are the same as the AFEs. During
259 training, the AFEs would change rapidly at first and slow down after several epochs. Therefore,
260 at the starting period of training, the same node would not be stably matched to the same set of
261 prototypes due to the rapidly changing AFEs. To avoid this from creating too many redundant
262 prototypes, we start to establish prototypes after training the AFEs at the 35th epochs. The input
263 data has a dimension of 1433, and we set the dimensions of A-, N-, and C-prototypes to be 16. The
264 number of training epochs is 90. Although the training procedure is designed in a delicate way,
265 the model is actually rather robust and can perform well without these delicate procedures. For
266 example, on the largest dataset OGB-Products, we only train the model for 10 epochs, and do not
267 decay the learning rate, the prototypes are established at the beginning, and the model still obtained
268 good results, as shown in the paper (e.g. results in Section 3.3). For the large datasets, for higher
269 efficiency, we set $h = 1$, and only uniformly sample one neighbor from the neighbors. On the
270 OGB-Products, we shrink the dimensions of A-, N-, and C-prototypes to be 2, in order to control the
271 number of prototypes. For both HPNs, the threshold t_A , t_N , and t_C are selected by cross validation
272 on $\{0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$. According to the experimental results, there is a
273 wide range for choosing the thresholds. Finally we choose $t_A = t_N = t_C = 0.3$.

274 The baselines have different settings. For the baselines with GCN backbone, 16 is approximately the
275 best for the number of hidden unit. For the GAT based baselines, we set the number of heads and

Table 2: Performance comparisons between HPNs and baseline models on four other datasets.

C.L.T.	Base	Actor		Wisc.		Corn.		Texas	
		AM	FM	AM	FM	AM	FM	AM	FM
None	GCN	43.63%	-9.11%	74.71%	-9.52%	34.92%	-68.00%	80.15%	-12.00%
	GAT	53.10%	-4.33%	78.82%	-4.76%	46.77%	-56.00%	74.62%	-16.00%
	GIN	45.51%	-8.88%	76.44%	-4.76%	34.92%	-64.00%	78.31%	-12.00%
EWC	GCN	44.29%	-7.06%	74.71%	-9.52%	38.92%	-60.00%	82.15%	-8.00%
	GAT	54.23%	-2.51%	78.82%	-4.76%	48.92%	-44.00%	78.62%	-8.00%
	GIN	47.61%	-7.29%	77.09%	0.00%	33.23%	-52.00%	78.31%	-12.00%
LwF	GCN	49.77%	-3.65%	84.65%	-9.52%	62.77%	-20.00%	56.31%	-52.00%
	GAT	52.82%	-6.15%	81.20%	0.00%	46.77%	-52.00%	78.46%	-20.00%
	GIN	49.70%	-4.10%	74.71%	0.00%	34.92%	-64.00%	34.92%	-52.00%
GEM	GCN	52.66%	+3.91%	88.71%	6.25%	65.08%	+0.00%	80.46%	+4.00%
	GAT	54.31%	-2.05%	77.09%	-9.52%	65.08%	-4.00%	76.77%	-4.00%
	GIN	45.23%	-11.16%	72.78%	-6.25%	76.62%	+4.00%	72.77%	+8.00%
MAS	GCN	50.73%	-1.59%	77.75%	-9.52%	61.23%	+0.00%	78.46%	+0.00%
	GAT	53.67%	-1.60%	76.01%	-6.25%	62.62%	-32.00%	84.46%	+0.00%
	GIN	51.69%	-0.69%	77.75%	-4.76%	63.08%	+0.00%	82.86%	+0.00%
ERGN.	GCN	52.44%	+0.69%	74.71%	-9.52%	34.92%	-68.00%	80.15%	-12.00%
	GAT	51.40%	-7.29%	78.16%	-9.52%	48.77%	-52.00%	80.46%	-12.00%
	GIN	42.72%	-12.98%	76.44%	-4.76%	34.92%	-64.00%	78.31%	-12.00%
TWP	GCN	50.59%	-4.79%	66.09%	-14.28%	56.77%	-32.00%	82.31%	-4.00%
	GAT	54.01%	-2.05%	80.54%	-9.52%	46.92%	-48.00%	78.62%	-8.00%
	GIN	49.91%	-3.64%	71.17%	-6.25%	51.08%	-24.00%	74.62%	-4.00%
Join.	GCN	57.01%	+0.00%	96.72%	+0.00%	88.09%	+0.00%	86.43%	+0.00%
	GAT	57.15%	+0.00%	95.94%	+0.00%	89.46%	+0.00%	86.30%	+0.00%
	GIN	56.97%	+0.00%	96.88%	+0.00%	88.82%	+0.00%	86.95%	+0.00%
HPNs		56.80%	-0.92%	96.55%	+0.00%	88.23%	+0.00%	86.31%	+2.77%

number of hidden units as 8. For GIN, the number of hidden units is 32. For all these baselines, the above mentioned settings are applied on most datasets. For some datasets on which the baselines cannot perform well, we will further tune the models carefully to get better results.

3 Additional Experimental Results and Detailed Analysis

To further validate our proposed model, in this section, we report additional experimental results by extending the experiments reported in the paper to more datasets. We will also give more detailed analysis on the results, which is omitted in the paper due to space limitations.

3.1 Comparisons with Baseline Methods on Additional Datasets

In this subsection, we include the comparison results with baseline methods on the other four datasets including Wisconsin, Cornell, Texas, and Actor, the results are shown in Table 2.

Among all the models in Table 1 of the paper, we could observe several kinds of forgetting behavior among the baselines. First, some of them are with low AM and severe forgetting (negative FM with large $|FM|$) like the pure GNNs in the first 3 rows of Table 1 in the paper. These models may perform well on individual tasks, but the AM is brought down by catastrophic forgetting. To explain this in details, we expand the results of GAT without any continual learning techniques in Table 3, in which Tr. denotes on which tasks has the model been trained and Te. denotes on which task the model is tested.

The first row of Table 3 shows the model performance on task 1 after it has been trained on the task 1, on the first two tasks, and on the first three tasks. We see that GAT performs well on each individual task it has just learnt. But after being trained on more tasks, the performance on previous tasks drops dramatically, making the AM to be relatively low. On the contrast, the performance change on previous tasks of HPNs is also shown in Table 3, and we could see that HPNs can well maintain the performance on previous tasks throughout the training on all tasks.

		GAT			HPNs		
Te. \ Tr.		\mathcal{T}_1	$\mathcal{T}_{1,2}$	$\mathcal{T}_{1,2,3}$	\mathcal{T}_1	$\mathcal{T}_{1,2}$	$\mathcal{T}_{1,2,3}$
\mathcal{T}_1		94.12%	47.96%	71.49%	95.02%	95.93%	94.57%
\mathcal{T}_2			93.30%	49.68%		92.66%	92.22%
\mathcal{T}_3				94.44%			96.83%

Table 3: Accuracy (%) changes of GAT and HPNs.

		GCN+GEM			GAT+MAS		
Te. \ Tr.		\mathcal{T}_1	$\mathcal{T}_{1,2}$	$\mathcal{T}_{1,2,3}$	\mathcal{T}_1	$\mathcal{T}_{1,2}$	$\mathcal{T}_{1,2,3}$
\mathcal{T}_1		93.67%	87.33%	76.92%	94.12%	90.50%	90.05%
\mathcal{T}_2			65.66%	69.33%		77.97%	70.84%
\mathcal{T}_3				80.95%			93.25%

Table 4: Accuracy (%) changes of GCN+GEM and GAT+MAS.

Second, there are also some models without severe forgetting but the still low AM. Typically these are the models which preserve the performance on previous tasks with certain constraints on the models. These constraints can indeed alleviate forgetting, but it also limits the model’s flexibility in learning new tasks, and thus the performance on new tasks will degrade. For example, comparing Table 3 with Table 4, GAT achieves accuracy higher than 93% on individual tasks, but for GAT+MAS, although the forgetting on task 1 is alleviated, the performance on task 2 drops significantly. GCN+GEM also suffers from the similar problem.

3.2 Additional Results on Ablation Study

In this subsection, we provide the ablation study results on another large dataset OGB-Arxiv, and the results are shown in Table 5 and 6.

From Table 5, we can observe that on the large dataset, the improvements brought by high level prototypes are more significant than on the small dataset (reported in Table 2 in the paper). Similarly, in Table 6, the influence of different loss terms is also more prominent compared to the results reported in Table 3 in the paper. The above results imply that our proposed hierarchical prototypes and different loss terms are effective, and the effectiveness becomes increasingly significant on larger datasets with richer information.

3.3 Additional Results on Learning Dynamics

Besides the learning dynamics on OGB-Arxiv provided in Section 3.5 in the paper, we further provide the results on OGB-Products, as shown in Figure 2 (Left).

The learning dynamics shown in Figure 2 (Left) is similar to the one on OGB-Products shown in the paper. The only difference is that OGB-Products contains more tasks and the ARS of the baselines decrease more than on OGB-Arxiv.

Table 5: Ablation study on prototypes of different levels of prototypes over OGB-Arxiv.

Conf.	A-p.	N-p.	C-p.	AM%	FM%
1	✓			82.1±0.9	+0.0±1.1
2	✓	✓		83.6±1.2	+0.2±0.9
3	✓	✓	✓	85.8±0.7	+0.6±0.9

Table 6: Ablation study on different loss terms over OGB-Arxiv.

Conf.	\mathcal{L}_{cls}	\mathcal{L}_{div}	\mathcal{L}_{dis}	AM%	FM%
1	✓			79.6±1.5	-0.3±1.3
2	✓	✓		82.3±1.0	+0.4±0.9
3	✓		✓	80.7±1.2	+0.0±1.4
4	✓	✓	✓	85.8±0.7	+0.6±0.9

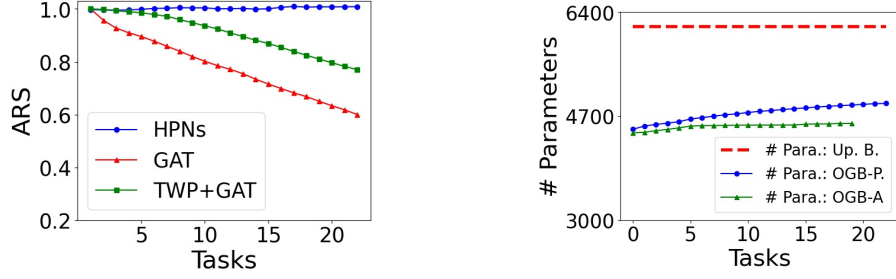


Figure 2: Left: dynamics of ARS for continual learning tasks on OGB-Products dataset. Right: dynamics of memory consumption of HPNs on both OGB-Arxiv and OGB-Products.

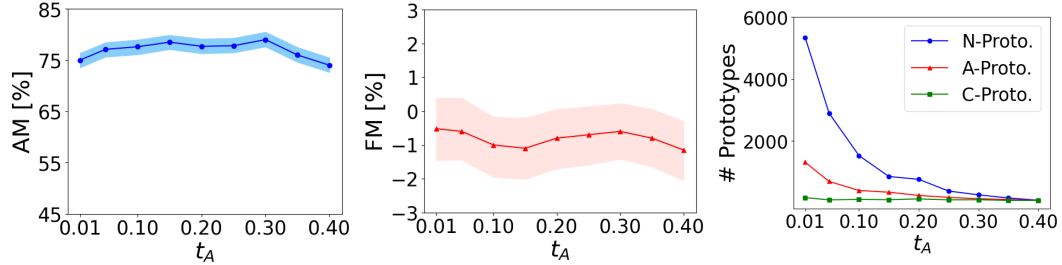


Figure 3: Left and Middle: AM and FM change when t_A varies on Citeseer. Right: impact of t_A on the number of prototypes over Citeseer.

3.4 Additional Results on Parameter Sensitivity

In Figure 3, we further provide the parameter sensitivity results on citeseer dataset. The results have similar patterns with the results provided in the paper.

3.5 Additional Results on Memory Consumption

In Figure 2 (Right), we simultaneously show the memory consumption change via the number of tasks on both OGB-Arxiv and OGB-Products. We use same model configurations for both datasets, thus the upper bounds of the memory consumption are same. From Figure 2 (Right), we could see that the memory consumption of HPNs on both datasets increases slowly and far less than the upper bound. Although OGB-Products is more than ten times larger than OGB-Arxiv, the memory used on OGB-Products is only slightly more than on OGB-Arxiv, demonstrating the memory efficiency of HPNs.

3.6 Additional Results on Visualization

In Figure 4, we visualize the hierarchical prototype representations of the test nodes on Citeseer by t-SNE [32]. Similar to the visualization results shown in the paper, Figure 4 sequentially show the classes of task 1 (Left), task 1,2 (Middle), and task 1,2,3 (Right). The examples belonging to different classes are denoted with different shapes and colors, as shown in the legend on the right.

4 Related Works

The proposed Hierarchical Prototype Networks (HPNs) are closely related to continual learning and graph representation learning. In this section, we provide more detailed discussions by comparing HPNs with the related works, especially on those directly applying existing continual learning methods on graph data.

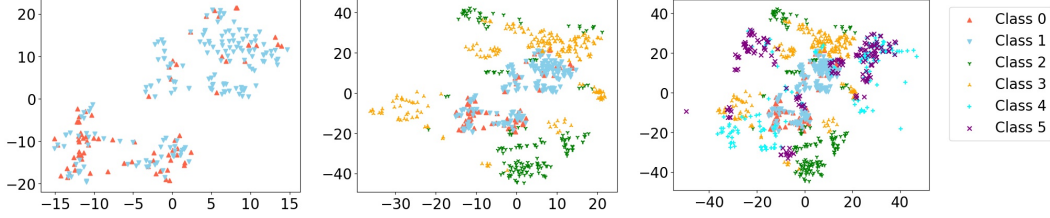


Figure 4: Visualization of hierarchical prototype representations of test data of different tasks from Citeseer via t-SNE.

4.1 Continual learning

Continual learning aims to overcome the well-known catastrophic forgetting problem that a model’s performance on previous tasks decreases significantly after being trained on new tasks. Existing works for continual learning can be categorized as regularization-based methods, memory-replay based methods, and parametric isolation based methods.

Regularization-based methods penalize the model objectives to maintain satisfactory performance on previous tasks [12, 17, 14, 8, 28]. For instance, Li and Hoiem [17] introduced Learning without Forgetting (LwF) which uses knowledge distillation to constrain the shift of parameters for old tasks; Kirkpatrick et al. [14] proposed elastic weight consolidation (EWC) that adds quadratic penalty to prevent the model weights from shifting too much. Recent works [8, 28] seek to constrain the gradients for new tasks in a subspace orthogonal to the updating directions that are important for previous tasks.

The regularization based methods are effective at preventing the forgetting on previous tasks. But the constraints on the model weights also limit a model’s capability to learn new tasks, and the overall performance will be affected.

Memory-replay based methods constantly feed a model with representative data of previous tasks to prevent forgetting [20, 30, 1, 4, 7]. One example is Gradient Episodic Memory (GEM) [20] that stores representative data in an episodic memory and add a constraint to prevent the loss on the episodic memory from increasing and only allows it to decrease. Instead of storing data, Shin et al. [30] added a generative model to generate pseudo-data of previous tasks to be interleaved with new task data for rehearsal. Recent works also look for better designs of memory replay so as to facilitate continual learning agents [4, 7].

Memory-replay based methods are currently one of the most effective approaches for alleviating catastrophic forgetting. But these methods are not suitable to be applied on graph data directly, as they are not designed to be capable of storing the topological information, which is a crucial part of graph data. In contrast, our proposed HPNs have explicitly designed modules for storing topological information. Moreover, memory-replayed methods require rehearsal of old data each time a new tasks is learnt, which induces extra computation burden.

Parametric isolation based methods adaptively introduce new parameters for new tasks to avoid the parameters of previous tasks being drastically changed [27, 41, 40, 37, 38]. For instance, progressive network [27] allocates a new sub-network for each new task and block any modification on the previously learnt networks. Yoon et al. [41] proposed a more flexible model (DEN) that dynamically adds new neurons for accommodating new tasks. Recently, various innovative approaches to allocate separated parameters for different tasks are proposed [38, 40, 37]. Besides the concrete models, a recent theoretical paper [15] analyzed the required capability of optimal continual learning agent.

Parametric isolation based methods are also effective at alleviating catastrophic forgetting, but the consistently increasing memory consumption could be a problem. Recently proposed methods have explicit mechanisms for controlling the memory consumption like merging parameters for similar tasks [40]. However, as the task specific parameters are fitted to each individual task, the parameter reuse level is limited. In our proposed HPNs, we consider the properties of graph data and decompose each node into a combinations of several prototypes. As the prototypes are shared by all tasks, the parameter reuse level is greatly increased, resulting in small memory consumption. Besides, we also have theoretical upper bound for the memory consumption.

Overall, although the existing continual learning methods can perform well on Euclidean data, they all have limitations when being applied onto graph data, while our proposed HPNs is specially designed for graph data without any of these limitations.

4.2 Graph representation learning

Graph representation learning aims to encode both the feature information from nodes and the topological structure of the incoming graph. Traditional methods relied on graph statistics or hand-crafted features [2, 18]. Recently, a great amount of attention has been paid to graph neural networks (GNNs), such as graph convolutional network (GCNs) [13], GraphSAGE [11], Graph Attention Networks (GATs) [33], and their extensions [39, 5, 46]. Instead of focusing on shallow networks, there are also works on building deep GCNs to further increase the capacity of GNNs [16, 6, 26, 43].

Currently, only limited efforts have been made to pursue continual graph representation. Zhou et al. [44] integrated memory-replay to GNNs by storing experience nodes from previous tasks. However, the topological structure of graphs is ignored. Liu et al. [19] developed topology-aware weight preserving (TWP) that can preserve the topological information of previous graphs. However, preserving topology of previous graphs will hamper its capability of learning topology on new graphs. Galke et al. [9] considers a scenario where new classes of nodes may appear, which is similar to our consideration. But they focus on adapting the model to new patterns and does not consider preserving the performance on previous tasks. [35] and [34] are also works related to both GNNs and continual learning. However, the setting of [35] is time step incremental, while our setting is class incremental. And in [34], the focus is on how to generalize the GNNs to networks with varying number of nodes, while the continual learning part is only applying existing memory replay method on the nodes.

Note that continual graph representation learning is essentially different from dynamic graphs which also deal with sequences of graphs [42, 24, 45, 21]. A key difference is that a dynamic graph sequence are status of a single graph at different time, while in continual learning setting a sequence contains different graphs for various tasks. Therefore, the methods developed for dynamic graphs cannot be directly applied to our tasks.

References

- [1] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pages 11816–11825, 2019.
- [2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.
- [3] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. The extreme classification repository: Multi-label datasets and code, 2016.
- [4] Lucas Caccia, Eugene Belilovsky, Massimo Caccia, and Joelle Pineau. Online learned continual compression with adaptive quantization modules. In *International Conference on Machine Learning*, pages 1240–1250. PMLR, 2020.
- [5] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.
- [6] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.
- [7] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, pages 1952–1961. PMLR, 2020.
- [8] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.
- [9] Lukas Galke, Iacopo Vagliano, and Ansgar Scherp. Incremental training of graph neural networks on temporal graphs under distribution shift. *arXiv preprint arXiv:2006.14422*, 2020.

- 434 [10] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing
435 system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.
- 436 [11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large
437 graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.
- 438 [12] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep
439 neural networks. *arXiv preprint arXiv:1607.00122*, 2016.
- 440 [13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional
441 networks. *arXiv preprint arXiv:1609.02907*, 2016.
- 442 [14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins,
443 Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al.
444 Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of
445 sciences*, 114(13):3521–3526, 2017.
- 446 [15] Jeremias Knoblauch, Hisham Husain, and Tom Diethe. Optimal continual learning has perfect
447 memory and is np-hard. In *International Conference on Machine Learning*, pages 5327–5337.
448 PMLR, 2020.
- 449 [16] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as
450 deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision*, pages
451 9267–9276, 2019.
- 452 [17] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern
453 analysis and machine intelligence*, 40(12):2935–2947, 2017.
- 454 [18] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks.
455 *Journal of the American society for information science and technology*, 58(7):1019–1031,
456 2007.
- 457 [19] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph
458 neural networks. *arXiv preprint arXiv:2012.06002*, 2020.
- 459 [20] David Lopez-Paz and Marc’ Aurelio Ranzato. Gradient episodic memory for continual learning.
460 In *Advances in neural information processing systems*, pages 6467–6476, 2017.
- 461 [21] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. Streaming graph neural
462 networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and
463 Development in Information Retrieval*, pages 719–728, 2020.
- 464 [22] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating
465 the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163,
466 2000.
- 467 [23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed
468 representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*,
469 2013.
- 470 [24] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunye Koh, and
471 Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of
472 the The Web Conference 2018*, pages 969–976, 2018.
- 473 [25] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn:
474 Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.
- 475 [26] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep
476 graph convolutional networks on node classification. In *International Conference on Learning
477 Representations*, 2019.
- 478 [27] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick,
479 Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv
480 preprint arXiv:1606.04671*, 2016.

- [28] Gobinda Saha and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representation*, 2021.
- [29] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.
- [30] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in neural information processing systems*, pages 2990–2999, 2017.
- [31] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816, 2009.
- [32] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [34] Chen Wang, Yuheng Qiu, and Sebastian Scherer. Lifelong graph learning. *arXiv preprint arXiv:2009.00647*, 2020.
- [35] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1515–1524, 2020.
- [36] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413, 2020.
- [37] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *arXiv preprint arXiv:2006.14769*, 2020.
- [38] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.
- [39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [40] Jaehong Yoon, Saehoon Kim, Eunho Yang, and Sung Ju Hwang. Scalable and order-robust continual learning with additive parameter decomposition. In *International Conference on Learning Representation*, 2020.
- [41] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.
- [42] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2672–2681, 2018.
- [43] Xikun Zhang, Chang Xu, and Dacheng Tao. On dropping clusters to regularize graph convolutional neural networks. 2020.
- [44] Fan Zhou, Chengtai Cao, Ting Zhong, Kunpeng Zhang, Goce Trajcevski, and Ji Geng. Continual graph learning. *arXiv preprint arXiv:2003.09908*, 2020.
- [45] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- [46] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. In *Advances in Neural Information Processing Systems*, pages 11247–11256, 2019.