A APPENDIX

A.1 SECOND-ORDER RANDOM WALKS (SECTION 2)

In second-order random walks, the probability of choosing v_{t+1} depends not only on v_t but also on v_{t-1}. We consider non-backtracking (Alon et al., 2007; Fitzner & van der Hofstad, 2013) and node2vec (Grover & Leskovec, 2016) random walks as representatives.

A non-backtracking random walk is defined upon a first-order random walk algorithm, e.g. one in Equation 4, by enforcing $v_{t+1} \neq v_{t-1}$:

$$\text{Prob}[v_{t+1} = x | v_t = j, v_{t-1} = i] := \begin{cases} \dfrac{\text{Prob}[v_{t+1} = x | v_t = j]}{\sum_{y \in N(j) \setminus \{i\}} \text{Prob}[v_{t+1} = y | v_t = j]} & \text{for } x \neq i, \\ 0 & \text{for } x = i, \end{cases} \tag{13}$$

where $\text{Prob}[v_{t+1} = x | v_t = j]$ is the probability of walking $j \to x$ given by the underlying first-order random walk. Notice that the probabilities are renormalized over $N(j) \setminus \{i\}$. This is ill-defined in the case the walk traverses $i \to j$ and reaches a dangling vertex j which has i as its only neighbor, since $N(j) \setminus \{i\} = \emptyset$. In such cases, we allow the random walk to "begrudgingly" backtrack (Rappaport et al., 2017), $i \to j$ and then $j \to i$, given that it is the only possible choice due to the dangling of j.

In case of node2vec random walk (Grover & Leskovec, 2016), a weighting term $\alpha(v_{t-1}, v_{t+1})$ with two (hyper)parameters, return p and in-out q, is introduced to modify the behavior of a first-order random walk:

$$\text{Prob}[v_{t+1} = x | v_t = j, v_{t-1} = i] := \frac{\alpha(i, x) \, \text{Prob}[v_{t+1} = x | v_t = j]}{\sum_{y \in N(j)} \alpha(i, y) \, \text{Prob}[v_{t+1} = y | v_t = j]}, \tag{14}$$

where the probability $\text{Prob}[v_{t+1} = x | v_t = j]$ is given by the underlying first-order random walk and $\alpha(v_{t-1}, v_{t+1})$ is defined as follows using the shortest path distance $d(v_{t-1}, v_{t+1})$:

$$\alpha(v_{t-1}, v_{t+1}) := \begin{cases} 1/p & \text{for } d(v_{t-1}, v_{t+1}) = 0, \\ 1 & \text{for } d(v_{t-1}, v_{t+1}) = 1, \\ 1/q & \text{for } d(v_{t-1}, v_{t+1}) = 2. \end{cases} \tag{15}$$

Choosing a large return parameter p reduces backtracking since it decreases the probability of walking from v_t to v_{t-1}. Choosing a small in-out parameter q has a similar effect of avoiding v_{t-1}, with a slight difference that it avoids the neighbors of v_{t-1} as well.

We now show an extension of Proposition 2.2 to the above second-order random walks:

Proposition A.1. *The non-backtracking random walk in Equation 13 and the node2vec random walk in Equation 14 are invariant if their underlying first-order random walk algorithm is invariant.*

Proof. The proof is given in Appendix A.9.3. □

A.2 MAIN ALGORITHM

We outline the main algorithms for the random walk and the recording function described in Section 2 and used in Section 5. Algorithm 1 shows the random walk algorithm, and Algorithms 2, 3, and 4 show the recording functions. For the random walk algorithm, we use non-backtracking described in Appendix A.1 and the minimum degree local rule conductance $c_G(\cdot)$ given in Equation 6 by default.

Based on the algorithms, in Figures 4, 5, and 6 we illustrate the task formats used for graph separation experiments in Section 5.2 by providing examples of input graphs, text records of random walks on them, and prediction targets for the language model that processes the records. Likewise, in Figures 7, 8, and 9 we show task formats for transductive classification on arXiv citation network in Section 5.3.

Algorithm 1: Random walk algorithm

Data: Input graph G, optional input vertex v, conductance function $c_G : E(G) \to \mathbb{R}_+$, walk length l, optional restart probability $\alpha \in (0, 1)$ or period $k > 1$
Result: Random walk $v_0 \to \cdots \to v_l$, restart flags $r_1, ..., r_l$

```
/* transition probabilities p : E(G) → ℝ₊ */
```
1 **for** $(u, x) \in E(G)$ **do**
2 $\quad p(u, x) \leftarrow c_G(u, x) / \sum_{y \in N(u)} c_G(u, y)$ // Equation 4
```
/* starting vertex v₀ */
```
3 **if** v *is given* **then**
```
   /* query starting vertex */
```
4 $\quad v_0 \leftarrow v$
5 **else**
```
   /* sample starting vertex */
```
6 $\quad v_0 \sim \text{Uniform}(V(G))$
```
/* random walk v₁ → ⋯ → vₗ, restart flags r₁,...,rₗ */
```
7 $v_1 \sim \text{Categorical}(\{p(v_0, x) : x \in N(v_0)\})$
8 $r_1 \leftarrow 0$
9 **for** $t \leftarrow 2$ **to** l **do**
```
   /* restart flag rₜ */
```
10 $\quad r_t \leftarrow 0$
11 $\quad$ **if** α *is given* **then**
```
      /* if restarted at t−1, do not restart */
```
12 $\quad\quad$ **if** $r_{t-1} = 0$ **then**
13 $\quad\quad\quad r_t \sim \text{Bernoulli}(\alpha)$
14 $\quad$ **else if** k *is given* **then**
15 $\quad\quad$ **if** $t \equiv 0 \pmod{k}$ **then**
16 $\quad\quad\quad r_t \leftarrow 1$
```
   /* random walk vₜ */
```
17 $\quad$ **if** $r_t = 0$ **then**
18 $\quad\quad S \leftarrow N(v_{t-1}) \setminus \{v_{t-2}\}$
19 $\quad\quad$ **if** $S \neq \emptyset$ **then**
```
         /* non-backtracking */
```
20 $\quad\quad\quad v_t \sim \text{Categorical}(\{p(v_{t-1}, x) / \sum_{y \in S} p(v_{t-1}, y) : x \in S\})$ // Equation 13
21 $\quad\quad$ **else**
```
         /* begrudgingly backtracking */
```
22 $\quad\quad\quad v_t \leftarrow v_{t-2}$
23 $\quad$ **else**
```
      /* restart */
```
24 $\quad\quad v_t \leftarrow v_0$

Algorithm 2: Recording function, using anonymization

Data: Random walk $v_0 \to \cdots \to v_l$, restart flags $r_1, ..., r_l$
Result: Text record $\mathbf{z}$

```
/* named vertices S, namespace id(·) */
```
1 $S \leftarrow \{v_0\}$
2 $\mathrm{id}(v_0) \leftarrow 1$
```
/* text record z */
```
3 $\mathbf{z} \leftarrow \mathrm{id}(v_0)$
4 for $t \leftarrow 1$ **to** l **do**
```
    /* anonymization vt ↦ id(vt) */
```
5   **if** $v_t \notin S$ **then**
6 $S \leftarrow S \cup \{v_t\}$
7 $\mathrm{id}(v_t) \leftarrow |S|$
```
    /* text record z */
```
8   **if** $r_t = 0$ **then**
```
        /* record walk vt−1 → vt */
```
9 $\mathbf{z} \leftarrow \mathbf{z} + \texttt{"-"} + \mathrm{id}(v_t)$
10   **else**
```
        /* record restart vt = v0 */
```
11 $\mathbf{z} \leftarrow \mathbf{z} + \texttt{";"} + \mathrm{id}(v_t)$

Algorithm 3: Recording function, using anonymization and named neighbors

---

Data: Random walk $v_0 \to \cdots \to v_l$, restart flags $r_1, ..., r_l$, input graph G
Result: Text record $\mathbf{z}$

```
/* named vertices S, namespace id(·), recorded edges T */
```
1 $S \leftarrow \{v_0\}$
2 $\mathrm{id}(v_0) \leftarrow 1$
3 $T \leftarrow \emptyset$
```
/* text record z */
```
4 $\mathbf{z} \leftarrow \mathrm{id}(v_0)$
5 for $t \leftarrow 1$ **to** l **do**
```
    /* anonymization vt ↦ id(vt) */
```
6   **if** $v_t \notin S$ **then**
7 $S \leftarrow S \cup \{v_t\}$
8 $\mathrm{id}(v_t) \leftarrow |S|$
```
    /* text record z */
```
9   **if** $r_t = 0$ **then**
```
        /* record walk vt−1 → vt */
```
10 $\mathbf{z} \leftarrow \mathbf{z} + \texttt{"-"} + \mathrm{id}(v_t)$
11 $T \leftarrow T \cup \{(v_{t-1}, v_t), (v_t, v_{t-1})\}$
```
        /* named neighbors U */
```
12 $U \leftarrow N(v_t) \cap S$
13   **if** $U \neq \emptyset$ **then**
```
            /* sort in ascending order id(u1) < ··· < id(u|U|) */
```
14 $[u_1, ..., u_{|U|}] \leftarrow \mathrm{SortByKey}(U, \mathrm{id})$
15   **for** $u \leftarrow u_1$ **to** $u_{|U|}$ **do**
```
                /* record named neighbors vt → u */
```
16   **if** $(v_t, u) \notin T$ **then**
17 $\mathbf{z} \leftarrow \mathbf{z} + \texttt{"#"} + \mathrm{id}(u)$
18 $T \leftarrow T \cup \{(v_t, u), (u, v_t)\}$

19   **else**
```
        /* record restart vt = v0 */
```
20 $\mathbf{z} \leftarrow \mathbf{z} + \texttt{";"} + \mathrm{id}(v_t)$

Algorithm 4: Recording function for transductive classification on arXiv network (Section 5.3)

Data: Directed input graph G, random walk $v_0 \to \cdots \to v_l$ and restart flags $r_1, ..., r_l$ on the undirected copy of G, paper titles $\{\mathbf{t}_v\}_{v \in V(G)}$ and abstracts $\{\mathbf{a}_v\}_{v \in V(G)}$, target category labels $\{\mathbf{y}_v\}_{v \in L}$ for labeled vertices $L \subset V(G)$

Result: Text record $\mathbf{z}$

```
/* named vertices S, namespace id(·), recorded edges T */
```
1   $S \leftarrow \{v_0\}$
2   $\text{id}(v_0) \leftarrow 1$
3   $T \leftarrow \emptyset$
```
/* text record z */
/* record starting vertex */
```
4   $\mathbf{z} \leftarrow$ `"Paper 1 - Title: {`$\mathbf{t}_{v_0}$`}", Abstract: {`$\mathbf{a}_{v_0}$`}"`
5   **for** $t \leftarrow 1$ **to** l **do**
```
      /* anonymization vt ↦ id(vt) */
```
6   **if** $v_t \notin S$ **then**
7      $S \leftarrow S \cup \{v_t\}$
8      $\text{id}(v_t) \leftarrow |S|$
```
      /* text record z */
```
9   **if** $r_t = 0$ **then**
```
         /* record walk vt−1 → vt with direction */
```
10   **if** $(v_{t-1}, v_t) \in E(G)$ **then**
11       $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\text{id}(v_{t-1})$`} cites Paper {`$\text{id}(v_t)$`}"`
12   **else**
13       $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\text{id}(v_{t-1})$`} is cited by Paper {`$\text{id}(v_t)$`}"`
14      $T \leftarrow T \cup \{(v_{t-1}, v_t), (v_t, v_{t-1})\}$
```
         /* record title tv, abstract av, and label yv if v is labeled */
```
15   **if** $\text{id}(v_t) = |S|$ **then**
16       $\mathbf{z} \leftarrow \mathbf{z} +$ `" - {`$\mathbf{t}_v$`}"`
17   **if** $v \in L$ **then**
18        $\mathbf{z} \leftarrow \mathbf{z} +$ `", Category: {`$\mathbf{y}_v$`}"`
19       $\mathbf{z} \leftarrow \mathbf{z} +$ `", Abstract: {`$\mathbf{a}_v$`}"`
20   **else**
21       $\mathbf{z} \leftarrow \mathbf{z} +$ `"."`
```
         /* named neighbors U */
```
22      $U \leftarrow N(v_t) \cap S$
23   **if** $U \neq \emptyset$ **then**
```
            /* sort in ascending order id(u1) < ··· < id(u|U|) */
```
24       $[u_1, ..., u_{|U|}] \leftarrow \text{SortByKey}(U, \text{id})$
25   **for** $u \leftarrow u_1$ **to** $u_{|U|}$ **do**
```
               /* record named neighbors vt → u with directions */
```
26   **if** $(v_t, u) \notin T$ **then**
27   **if** $(v_t, u) \in E(G)$ **then**
28          $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\text{id}(v_t)$`} cites Paper {`$\text{id}(u)$`}."`
29   **else**
30          $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\text{id}(v_t)$`} is cited by Paper {`$\text{id}(u)$`}."`
31        $T \leftarrow T \cup \{(v_t, u), (u, v_t)\}$

32   **else**
```
         /* record restart */
```
33      $\mathbf{z} \leftarrow \mathbf{z} +$ `" Restart at Paper 1."`

Record	[CLS]1-2-3-4-5-6-7-8-9#6-10#5-11-12-13#1-14-15#1#3-16-17#3#5-5-10-11-18#9#13-13-14-19#16-16-17-5-6-20#4-21-22-23-24#2-25#1#8#18-8-26#24-27#23-28#19#22-22-29#19-19-14-13-18-11-10-5-17-30#10-31#16#29-29-32#21-33#7#20-7-34#26-35#27-36#12#14#28-28-22-29-31-37#32-32-33-7-6-5-4-20-6-5-4-20-33-32-37-38#11#30-39#12#35-12-11-10-5-4-40#2#21#23-2-24-23-27-28-19-16-17-30-10-5-4-40-2-3-4-5-17-3-2-40-4-20-33-32-21-20-33-41#34#37#39-39-38-37-31-29-32-33-41-34-35-36-28-27-35-39-12-13-1-15-3-4-20-6-7-33-41-34-35-39-41-37-32-33-20-21-40-23-22-29-31-37-41-34-35-39-12-13-18-25-24-26-34-7-33-20-4-40-23-27-35-36-12-11-10-30-31-16-15-3-4-40-23-22-28-36-14-19-29-31-37-41-33-32-29-22-21-32-33-20-4-40-21-22-29-19-14-15-16-31-29-[SEP]
Answer	csl(41, 9)

Record	[CLS]1-2-3-4-5-6-7-8-9-10-11-12-13-14#8-15-16#6#8-6-7-8-14-13-17-18#12-19#1-20#17-21-22-23#4-24#21-21-20-17-13-25#15-26-27#5#7-28#12#25-12-13-14-29#9#17-17-13-25-28-12-13-14-8-9-10-30#20#22#29-22-31#10-32#1#11#18-1-19-33#2#21-21-22-31-32-1-19-33-2-34-35#15#26-36#16-16-15-25-28-27-7-6-37#4#24#36-4-23-24-38#33#34#36-34-2-1-39#3#23#31-23-24-38-33-19-1-2-3-39-23-24-21-33-38-24-23-4-37-24-23-39-1-2-34-38-36-37-6-5-4-23-39-3-4-5-6-16-15-35-26-40#3#5#34-5-6-7-27-28-12-13-17-29-9-41#7#11#28-7-27-26-25-13-12-11-32-18-12-13-14-15-25-26-40-34-38-36-37-24-23-4-37-24-21-33-2-3-4-5-27-7-6-16-15-35-34-2-3-40-26-35-34-38-33-21-20-17-29-14-15-16-8-9-41-28-27-7-8-14-15-25-28-27-5-6-7-41-9-8-7-41-9-10-31-[SEP]
Answer	csl(41, 16)

Figure 4: Two CSL graphs and text records of random walks from Algorithms 1 and 3. Task is graph classification into 10 isomorphism types $\text{csl}(41, s)$ for skip length $s \in \{2, 3, 4, 5, 6, 9, 11, 12, 13, 16\}$. In random walks, we label vertices by anonymization and color edges by their time of discovery.

Graph	Random Walk

Record   [CLS]1-2-3-4#2-5-6-7#1#2#5-8#3#5#6-9#3-3-4-5-10#1#4-11#1#3#8#9-12#1#6#10-13#6#9-6-5-8-7-2-14#3#4#6#12#13-13-9-15#1#2#7#13-7-6-13-12-1-2-4-10-12-11-1-7-15-1-11-12-10-4-14-13-15-2-7-8-9-3-2-7-8-5-7-2-3-14-13-15-1-2-14-13-16#4#5#9#10#15-5-10-1-11-3-14-4-16-10-1-12-13-6-12-13-6-12-10-11-12-10-11-1-7-6-13-15-16-5-4-14-3-9-13-16-15-1-12-13-9-15-7-2-14-13-6-5-7-2-4-3-9-15-13-14-6-5-7-8-9-11-10-5-16-4-3-14-2-15-13-12-10-5-16-4-14-13-12-14-13-16-10-5-6-14-3-9-16-13-9-15-1-2-15-7-1-12-13-6-8-5-7-6-12-1-11-12-14-4-2-15-9-8-7-1-15-16-4-14-12-1-10-5-16-9-8-5-16-13-12-6-14-2-1-7-15-16-4-2-1-15-16-10-4-16-15-9-16-15-13-6-8-9-16-5-6-[SEP]

Answer   4 × 4 rook's graph

Graph	Random Walk

Record   [CLS]1-2-3-4#2-5#3-6#1-7#5-8#3#5-9#3-3-4-5-10#1#4#6-11#4#9-12#2#4#7-13#6#7#9#11-7-14#1#2#8#12-8-5-6-7-13-9-11-4-5-3-4-12-14-1-10-6-15#1#2#3#9#13-2-14-16#1#8#9#10#11-10-11-4-10-6-1-16-8-7-5-8-9-3-5-7-8-14-7-6-15-2-3-8-14-2-1-16-11-4-10-6-15-13-6-1-16-10-6-1-16-14-12-13-7-12-4-3-2-12-11-4-5-7-13-15-3-5-6-1-15-9-13-11-16-10-6-13-9-11-4-12-7-13-6-7-5-6-1-15-9-11-13-6-5-4-2-3-9-15-1-14-16-1-15-6-7-13-15-1-10-5-3-2-1-16-10-4-11-16-10-5-4-12-11-9-16-11-9-13-12-14-16-1-10-4-11-12-13-6-5-7-12-2-3-5-4-10-6-13-9-8-5-10-16-11-12-2-1-6-10-5-3-9-8-14-16-11-10-5-6-7-12-14-16-11-12-14-1-16-11-10-4-11-13-9-16-8-3-5-8-9-16-14-2-[SEP]

Answer   Shrikhande graph

Figure 5: SR16 graphs and text records of random walks from Algorithms 1 and 3. The task is graph classification into two isomorphism types {4 × 4 rook's graph, Shrikhande graph}. In random walks, we label vertices by anonymization and color edges by their time of discovery.

Record	[CLS]1-2-3#1-4#2-5#1#2-6#4-7#1#3-3-8#1#4#5#7-5-9#1#2#7-10#4#7#8-11#1#4#8#9-8-12#1# 3#6#7-3-13#2#4#6#7#9#11-2-3-7-6-14#2#5#7#8#13-15#2#3#7#9#10-16#3#5#8#10#14-17#1#6# 7#10#11#13#14-13-18#5#8#9#11#12#14#16-19#3#6#9#11#12#13#15#16-9-7-13-11-20#1#2#12# 14#15#17#18-21#2#4#6#8#10#11#12#14#15#19-8-7-14-5-22#4#6#7#9#10#12#15#18#20-23#2# 3#4#10#12#13#16#17#18#20-13-9-2-21-24#1#2#5#6#9#10#12#16#17#19#23-19-6-24-23-16-3- 1-5-8-21-14-2-3-7-8-5-6-7-1-25#3#4#5#6#11#15#16#17#19#20#22-17-1-11-4-3-15-20-2-5- 22-12-21-14-7-9-5-4-22-20-25-11-21-8-1-24-17-10-23-18-13-6-4-8-3-7-17-14-20-12-24- 16-17-10-21-14-5-4-13-14-7-13-3-8-1-3-7-14-13-2-3-8-18-22-5-[SEP]
Answer	Graph 1

Record	[CLS]1-2-3#1-4#1-5#2-6#2#4-7#2#3#4-3-8#4#5#7-9#3#6-10#1#6-11#1#4#6#7#8-12#2#7#8#10- 8-13#7#9#12-14#1#2#3#11#12-15#1#2#3#5#6#8#9#10#13-1-3-16#1#4#7#9#10#13-13-17#1#4# 6#9#11#14-18#2#3#6#7#9#10#12-19#3#5#8#9#11#14#17-11-7-20#5#6#10#11#13#14#15#16#19- 21#1#2#5#7#13#16#17#18#19-16-13-15-22#1#4#5#8#10#12#13#17#18#21-10-6-18-7-11-23#1# 2#5#8#9#10#12#16#19#21-12-8-4-24#2#5#6#9#12#13#14#16#17#23-16-10-23-9-6-5-8-19-3-1- 23-8-22-5-2-3-16-25#3#4#5#10#12#14#18#19#20#22#24-14-13-16-1-21-17-1-22-15-3-9-13- 16-21-22-15-10-11-4-24-9-6-10-12-14-24-12-8-7-11-20-5-25-20-7-6-4-8-3-16-21-20-13- 15-5-19-23-5-6-17-9-6-7-11-4-7-3-8-7-3-16-23-2-1-3-8-5-24-23-[SEP]
Answer	Graph 8

Figure 6: Two SR25 graphs and text records of random walks from Algorithms 1 and 3. The task is graph classification into 15 isomorphism types. In random walks, we label vertices by anonymization and color edges by their time of discovery.

System	A walk on the arXiv citation network will be given. Predict the arXiv CS sub-category Paper 1 belongs to. It is one of the following: Numerical Analysis (cs.NA), Multimedia (cs.MM), Logic in Computer Science (cs.LO), ... Discrete Mathematics (cs.DM). Only respond with the answer, do not say any word or explain.
Record	A walk on arXiv citation network is as follows: Paper 1 – Title: Safety guided deep reinforcement learning via online gaussian process estimation, Abstract: An important facet of reinforcement learning (rl) has to do with how the agent goes about exploring the environment. traditional exploration strategies typically focus on efficiency and ignore safety. however, for practical applications, ensuring safety of the agent during exploration is crucial since performing an unsafe action or reaching an unsafe state could result in irreversible damage to the agent. the main challenge of safe exploration is that characterizing the unsafe states and actions... Restart at 1. Paper 1 is cited by 2 – Learning to walk in the real world with minimal human effort, Abstract: Reliable and stable locomotion has been one of the most fundamental challenges for legged robots. deep reinforcement learning (deep rl) has emerged as a promising method for developing such control policies... Restart at 1. Paper 1 cites 3 – Deep q learning from demonstrations, Category: Artificial Intelligence (cs.AI), Abstract: Deep reinforcement learning (rl) has achieved several high profile successes in difficult decision-making problems. however, these algorithms typically require a huge amount of data before they reach... Restart at 1. Paper 1 cites 4 – Constrained policy optimization, Category: Machine Learning (cs.LG), Abstract: For many applications of reinforcement learning it can be more convenient to specify both a reward function and constraints, rather than trying to design behavior through the reward function. for example,... Paper 4 is cited by 2. Paper 4 is cited by 5 – Artificial intelligence values and alignment, Abstract: This paper looks at philosophical questions that arise in the context of ai alignment. it defends three propositions. first, normative and technical aspects of the ai alignment problem are interrelated,... Paper 5 cites 6 – Agi safety literature review, Category: Artificial Intelligence (cs.AI), Abstract: The development of artificial general intelligence (agi) promises to be a major event. along with its many potential benefits, it also raises serious safety concerns (bostrom, 2014). the intention of... Paper 6 is cited by 7 – Modeling agi safety frameworks with causal influence diagrams, Abstract: Proposals for safe agi systems are typically made at the level of frameworks, specifying how the components of the proposed system should be trained and interact with each other. in this paper, we model... Restart at 1. Paper 1 cites 8 – Safe learning of regions of attraction for uncertain nonlinear systems with gaussian processes, Category: Systems and Control (cs.SY), Abstract: Control theory can provide useful insights into the properties of controlled, dynamic systems. one important property of nonlinear systems is the region of attraction (roa), a safe subset of the state... Paper 8 is cited by 9 – Control theory meets pomdps a hybrid systems approach, Abstract: Partially observable markov decision processes (pomdps) provide a modeling framework for a variety of sequential decision making under uncertainty scenarios in artificial intelligence (ai). since the... Restart at 1.

...

Which arXiv CS sub-category does Paper 1 belong to? Only respond with the answer, do not say any word or explain. |
| Answer | Machine Learning (cs.LG) |

Figure 7: Transductive classification on arXiv citation network using text record of random walk from Algorithms 1 and 4. Colors indicate task instruction, walk information, and label information.

System	Title and abstract of an arXiv paper will be given. Predict the arXiv CS sub-category the paper belongs to. It is one of the following: Numerical Analysis (cs.NA), Multimedia (cs.MM), Logic in Computer Science (cs.LO), ... Discrete Mathematics (cs.DM). Only respond with the answer, do not say any word or explain.
Context	Title: Safety guided deep reinforcement learning via online gaussian process estimation Abstract: An important facet of reinforcement learning (rl) has to do with how the agent goes about exploring the environment. traditional exploration strategies typically focus on efficiency and ignore safety. however, for practical applications, ensuring safety of the agent during exploration is crucial since performing an unsafe action or reaching an unsafe state could result in irreversible damage to the agent. the main challenge of safe exploration is that characterizing the unsafe states and actions... Which arXiv CS sub-category does this paper belong to?
Answer	Machine Learning (cs.LG)

Figure 8: Zero-shot format for vertex classification on arXiv citation network. Task instruction and label information are colored.

System	Title and abstract of an arXiv paper will be given. Predict the arXiv CS sub-category the paper belongs to. It is one of the following: Numerical Analysis (cs.NA), Multimedia (cs.MM), Logic in Computer Science (cs.LO), ... Discrete Mathematics (cs.DM).
Context	Title: Deeptrack learning discriminative feature representations online for robust visual tracking Abstract: Deep neural networks, albeit their great success on feature learning in various computer vision tasks, are usually considered as impractical for online visual tracking, because they require very long... Category: cs.CV Title: Perceived audiovisual quality modelling based on decison trees genetic programming and neural networks... Abstract: Our objective is to build machine learning based models that predict audiovisual quality directly from a set of correlated parameters that are extracted from a target quality dataset. we have used the... Category: cs.MM ... Title: Safety guided deep reinforcement learning via online gaussian process estimation Abstract: An important facet of reinforcement learning (rl) has to do with how the agent goes about exploring the environment. traditional exploration strategies typically focus on efficiency and ignore safety. however, for practical applications, ensuring safety of the agent during exploration is crucial since performing an unsafe action or reaching an unsafe state could result in irreversible damage to the agent. the main challenge of safe exploration is that characterizing the unsafe states and actions... Which arXiv CS sub-category does this paper belong to?
Answer	cs.LG

Figure 9: One-shot format for transductive classification on arXiv citation network. 40 labeled examples are given in form of multi-turn dialogue. Task instruction and label information are colored.

```
[...]  A walk on arXiv citation network is as follows:
Paper 1 - Title: Unsupervised and interpretable scene discovery with discrete attend infer
repeat, Abstract: In this work we present discrete attend infer repeat (discrete-air), a
recurrent auto-encoder with structured latent distributions containing discrete categori
cal distributions, continuous attribute distributions, and factorised spatial attention.
while inspired by the original air model andretaining air model's capability in identify
ing objects in an image, discrete-air provides direct interpretability of the latent codes.
we show that for multi-mnist and a multiple-objects version of dsprites... Restart at 1.
[...]  Restart at 1. Paper 1 cites 12 - Attend infer repeat fast scene understanding with
generative models, Category: Computer Vision and Pattern Recognition (cs.CV), Abstract: We
present a framework for efficient inference in structured image models that explicitly rea
son about objects. we achieve this by performing probabilistic inference using a recurrent
neural network that... Paper 12 cites 3. Paper 12 cites 7. Paper 12 is cited by 9. Paper
12 is cited by 10. Paper 12 cites 11. Restart at 1. Paper 1 cites 11. Restart at 1. Paper
1 cites 12. Paper 12 is cited by 13 - An architecture for deep hierarchical generative mod
els, Category: Machine Learning (cs.LG), Abstract: We present an architecture which lets us
train deep, directed generative models with many layers of latent variables. we include de
terministic paths between all latent variables and the generated output,... Paper 13 cites
3. Restart at 1.  [...]
                              Attention (Layer 13/30)
```

Answer	cs.CV
Prediction	cs.CV

Figure 10: Example of attention in a frozen Llama 3 8B applied on arXiv transductive classification. Attention weights are averaged over all 30 heads.

A.3 ATTENTION VISUALIZATIONS

In Figure 10, we show an example of self-attention in the frozen Llama 3 8B model applied to arXiv transductive classification (Section 5.3). We show attention weights on text record of random walk from the generated cs token as query, which is just before finishing the prediction, e.g., cs.CV. We color the strongest activation with orange, and do not color values below 1% of it. The model invests a nontrivial amount of attention on walk information such as Paper 1 cites 12, while also making use of titles and labels of labeled vertices. This indicates the model is utilizing the graph structure recorded by the random walk in conjunction with other information to make predictions.

In Figures 11, 12, and 13, we show examples of self-attention in DeBERTa models trained for graph separation (Section 5.2). We first show attention weights on text record of random walk from the [CLS] query token. Then, we show an alternative visualization where the attention weights are mapped onto the original input graph. For example, for each attention on v_{t+1} where the walk has traversed $v_t \rightarrow v_{t+1}$, we regard it as an attention on the edge (v_t, v_{t+1}) in the input graph. We ignore [CLS] and [SEP] tokens since they do not appear on the input graph. We observe that the models often focus on sparse, connected substructures, which presumably provide discriminative information on the isomorphism types. In particular, for CSL graphs (Figure 11) we observe an interpretable pattern of approximate cycles composed of skip links. This is presumably related to measuring the lengths of skip links, which provides sufficient information of isomorphism types of CSL graphs.

```
1-2-3-4-5-6-7-8#5-9#1#4-1-10-11#2-2-1-12#3#8-3-4-9-8-5-13-14#7-15#6-16-17-18-19#14#16-14-7-
8-12-1-9-20#3#10-21#2-22#11-23-24#10#21-10-1-12-25#4#7#13-7-6-5-4-25-12-8-5-4-25-12-8-7-25-
13-26#6#19-27#15#18-18-17-28-29-30-31-32#28-33#29-34#23#31-31-32-35#17#30-17-16-19-14-15-6-
7-8-5-6-7-8-9-4-3-12-1-10-20-9-8-7-14-19-16-15-6-5-4-9-8-5-4-9-20-21-22-11-36#24-37#23#31-
31-34-33-32-28-38#16#27-27-15-16-17-18-27-26-13-25-7-14-19-26-6-15-16-38-27-18-17-35-30-31-
34-23-24-21-2-1-10-11-22-39#34#36-40#30#33#37-33-29-30-35-32-33-40-30-35-32-31-37-36-39-34-
31-30-29-41#18#35#38-18-27-38-16-17-28-38-27-26-13-14-15-27-26-13-14-15-6-26-13-5-6-15-14-
13-25-7-14-15-6-26-27-38-28-29-33-34-39-36-24-10-1-2-
                          Attention (Head 12/12, Layer 12/12)
```


Graph   Attention Mapped onto Graph

```
1-2-3-4-5-6#3-7-8-9#1-10#7-7-11#5-5-6-3-4-12-13#5-14#11-15-16#8#11-11-5-13-12-4-5-6-3-4-12-
17#8-18#16-19#15-20-21-22-23#1-24-25#1#10-26-27-28-29#24#26-24-25-10-9-1-25-24-30#20#22-31#
19#29-29-28-27-32#22-33#21#28-21-34#13-35#12#18#20-12-17-36#2#4#9-9-10-25-26-27-32-22-23-24-
25-10-9-8-16-18-19-20-21-34-37#14#33-38#15#28#31-15-14-13-34-21-22-32-39#2#23-2-1-23-24-30-
31-38-15-19-31-30-24-29-31-38-15-19-20-29-31-38-15-19-14-37-34-35-20-21-34-37-33-32-22-21-34-
37-33-32-22-30-20-35-18-16-15-14-13-12-35-20-30-22-32-33-37-14-11-5-4-36-2-1-25-26-29-24-23-
39-32-22-23-24-25-1-2-36-9-10-25-24-30-31-19-15-16-11-14-37-38-15-14-11-7-10-40#6#26-6-7-10-
9-8-17-18-16-11-7-8-16-11-7-8-17-18-16-15-38-37-14-11-5-4-36-17-18-
                          Attention (Head 5/12, Layer 12/12)
```


Graph   Attention Mapped onto Graph

Figure 11: Examples of attention from [CLS] token in a DeBERTa trained on CSL graph separation, forming approximate cycles using skip links. This is presumably related to measuring the lengths of skip links, which provides sufficient information to infer isomorphism types of CSL graphs.

28

1-2-3-4#2-5-6-7#1#2#5-8#3#5#6-9#3-3-4-5-10#1#4-11#1#3#8#9-12#1#6#10-13#6#9-6-5-8-7-2-14#3#
4#6#12#13-13-9-15#1#2#7#13-7-6-13-12-1-2-4-10-12-11-1-7-15-1-11-12-10-4-14-13-15-2-7-8-9-3-
2-7-8-5-7-2-3-14-13-15-1-2-14-13-16#4#5#9#10#15-5-10-1-11-3-14-4-16-10-1-12-13-6-12-13-6-12-
10-11-12-10-11-1-7-6-13-15-16-5-4-14-3-9-13-16-15-1-12-13-9-15-7-2-14-13-6-5-7-2-4-3-9-15-
13-14-6-5-7-8-9-11-10-5-16-4-3-14-2-15-13-12-10-5-16-4-14-13-12-14-13-16-10-5-6-14-3-9-16-
13-9-15-1-2-15-7-1-12-13-6-8-5-7-6-12-1-11-12-14-4-2-15-9-8-7-1-15-16-4-14-12-1-10-5-16-9-8-
5-16-13-12-6-14-2-1-7-15-16-4-2-1-15-16-10-4-16-15-9-16-15-13-6-8-9-16-5-6-

Attention (Head 11/12, Layer 10/12)

Graph

Attention Mapped onto Graph

1-2-3-4#2-5#3-6#1-7#5-8#3#5-9#3-3-4-5-10#1#4#6-11#4#9-12#2#4#7-13#6#7#9#11-7-14#1#2#8#12-8-
5-6-7-13-9-11-4-5-3-4-12-14-1-10-6-15#1#2#3#9#13-2-14-16#1#8#9#10#11-10-11-4-10-6-1-16-8-7-
5-8-9-3-5-7-8-14-7-6-15-2-3-8-14-2-1-16-11-4-10-6-15-13-6-1-16-10-6-1-16-14-12-13-7-12-4-3-
2-12-11-4-5-7-13-15-3-5-6-1-15-9-13-11-16-10-6-13-9-11-4-12-7-13-6-7-5-6-1-15-9-11-13-6-5-4-
2-3-9-15-1-14-16-1-15-6-7-13-15-1-10-5-3-2-1-10-4-11-16-10-5-4-12-11-9-16-11-9-13-12-14-
16-1-10-4-11-12-13-6-5-7-12-2-3-5-4-10-6-13-9-8-5-10-16-11-12-2-1-6-10-5-3-9-8-14-16-11-10-
5-6-7-12-14-16-11-12-14-1-16-11-10-4-11-13-9-16-8-3-5-8-9-16-14-2-

Attention (Head 11/12, Layer 7/12)

Graph

Attention Mapped onto Graph

Figure 12: Examples of attention from [CLS] token in a DeBERTa trained on SR16 graph separation. The model tend to focus on named neighbor records that form a sparse connected substructure, which we conjecture to provide discriminative information on the isomorphism type of SR16 graphs.

1-2-3#1-4#2-5#1#2-6#4-7#1#3-3-8#1#4#5#7-5-9#1#2#7-10#4#7#8-11#1#4#8#9-8-12#1#3#6#7-3-13#2#
4#6#7#9#11-2-3-7-6-14#2#5#7#8#13-15#2#3#7#9#10-16#3#5#8#10#14-17#1#6#7#10#11#13#14-13-18#
5#8#9#11#12#14#16-19#3#6#9#11#12#13#15#16-9-7-13-11-20#1#2#12#14#15#17#18-21#2#4#6#8#10#11#
12#14#15#19-8-7-14-5-22#4#6#7#9#10#12#15#18#20-23#2#3#4#10#12#13#16#17#18#20-13-9-2-21-24#
1#2#5#6#9#10#12#16#17#19#23-19-6-24-23-16-3-1-5-8-21-14-2-3-7-8-5-6-7-1-25#3#4#5#6#11#15#
16#17#19#20#22-17-1-11-4-3-15-20-2-5-22-12-21-14-7-9-5-4-22-20-25-11-21-8-1-24-17-10-23-18-
13-6-4-8-3-7-17-14-20-12-24-16-17-10-21-14-5-4-13-14-7-13-3-8-1-3-7-14-13-2-3-8-18-22-5-

Attention (Head 4/12, Layer 9/12)

Graph   Attention Mapped onto Graph

1-2-3#1-4#1-5#2-6#2#4-7#2#3#4-5-8#4#5#7-9#3#6-10#1#6-11#1#4#6#7#8-12#2#7#8#10-8-13#7#9#12-
14#1#2#3#11#12-15#1#2#3#5#6#8#9#10#13-1-3-16#1#4#7#9#10#13-13-17#1#4#6#9#11#14-18#2#3#6#7#
9#10#12-19#3#5#8#9#11#14#17-11-7-20#5#6#10#11#13#14#15#16#19-21#1#2#5#7#13#16#17#18#19-15-
13-15-22#1#4#5#8#10#12#13#17#18#21-10-6-18-7-11-23#1#2#5#8#9#10#12#16#19#21-12-8-4-24#2#5#
6#9#12#13#14#16#17#23-16-10-23-9-6-5-8-19-3-1-23-8-22-5-2-3-16-25#3#4#5#10#12#14#18#19#20#
22#24-14-13-16-1-21-17-1-22-15-3-9-13-16-21-22-15-10-11-4-24-9-6-10-12-14-24-12-8-7-11-20-5-
25-20-7-6-4-8-3-16-21-20-13-15-5-19-23-5-6-17-9-6-7-11-4-7-3-8-7-3-16-23-2-1-3-8-5-24-23-

Attention (Head 10/12, Layer 8/12)

Graph   Attention Mapped onto Graph

Figure 13: Examples of attention from [CLS] token in a DeBERTa trained on SR25 graph separation. The model tend to focus on named neighbor records that form a sparse connected substructure, which we conjecture to provide discriminative information on the isomorphism type of SR25 graphs.

Table 5: Over-smoothing and over-squashing results, extending Table 2 (Section 5.1). We report test MSE aggregated for 4 randomized runs, except for CRaWl which took >3 days per run.

Method	Clique (over-smoothing)	Barbell (over-squashing)
GCN	29.65 ± 0.34	1.05 ± 0.08
SAGE	0.86 ± 0.10	0.90 ± 0.29
GAT	20.97 ± 0.40	1.07 ± 0.09
NSD	0.08 ± 0.02	1.09 ± 0.15
BuNN	0.03 ± 0.01	0.01 ± 0.07
$l = 100$		
RWNN-MLP	0.082 ± 0.018	0.326 ± 0.035
WalkLM-transformer	0.096 ± 0.018	0.388 ± 0.048
RWNN-transformer (Ours)	0.049 ± 0.007	0.297 ± 0.050
$l = 1000$		
CRaWl	0.103 ± N/A	0.289 ± N/A
CRaWl*	0.121 ± N/A	0.103 ± N/A
RWNN-MLP	0.028 ± 0.026	0.015 ± 0.005
WalkLM-transformer	**0.023 ± 0.003**	0.037 ± 0.009
RWNN-transformer (Ours)	**0.023 ± 0.006**	**0.007 ± 0.001**

A.4 SUPPLEMENTARY MATERIALS FOR SYNTHETIC EXPERIMENTS (SECTION 5.1)

We provide experimental details and additional baseline comparisons for the Clique and Barbell experiments in Section 5.1 we could not include in the main text due to space constraints.

Dataset In Clique and Barbell (Bamberger et al., 2024), each vertex of a graph G is given a random scalar value. The task at each vertex v is to regress the mean of all vertices with an opposite sign. In Clique, G is a 20-clique, and its vertices are randomly partitioned into two sets of size 10, and one set is given positive values and the other is given negative values. In Barbell, two 10-cliques are linked to form G and one clique is given positive values and the other is given negative values.

Models Our RWNN uses MDLR walks with non-backtracking, and since the task is vertex-level, for each v we fix the starting vertex as $v_0 = v$. The recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ takes a walk and produces a record $\mathbf{z} := (\mathrm{id}(v_0), \mathbf{x}_{v_0}, \mathbf{x}_{v_0}) \to \cdots \to (\mathrm{id}(v_l), \mathbf{x}_{v_l}, \mathbf{x}_{v_0})$, where $\mathrm{id}(v)$ is the anonymization of v and $\mathbf{x}_v$ is the scalar value of v. We find that recording $\mathbf{x}_{v_0}$ at each step improves training. The reader NN is a 1-layer transformer encoder, with 128 feature dimensions, sinusoidal positional embedding, and average pooling readout. To map the recording at each step $(\mathrm{id}(v_t), \mathbf{x}_{v_t}, \mathbf{x}_{v_0})$ to a 128-dimensional feature, we use a learnable vocabulary for $\mathrm{id}(v_t)$, a learnable linear layer for $\mathbf{x}_{v_t}$, and a 3-layer MLP with 32 hidden dimensions and tanh activation for $\mathbf{x}_{v_0}$.

We test additional baselines that use random walks. We test CRaWl (Tönshoff et al., 2023) using the official code, and only remove batch normalization since it led to severe overfitting for Clique and Barbell. Since CRaWl uses vertex-level average pooling, while RWNN-transformer uses walk-level, we also test a variant that uses walk-level pooling and denote it by CRaWl*. We also test RWNN-MLP, which uses an MLP on the concatenated walk features instead of a transformer. The MLP has 3 layers with 128 hidden dimensions and ReLU activation. We also test an adaptation of WalkLM (Tan et al., 2023). While the work entails two ideas, representation of walks and fine-tuning by masked auto-encoding, we focus on testing its choice of walks and their representation since the fine-tuning method is orthogonal to our work. We do this by modifying RWNN-transformer to use uniform walks and omit anonymization $\mathrm{id}(v_t)$ from the walk record, and denote the model by WalkLM-transformer.

We train all models with Adam optimizer with batch size 1 and learning rate 1e-4 for 500 epochs, closely following Bamberger et al. (2024) except for the lower learning rate which we found necessary for stabilizing training due to the stochasticity of walk based methods. For all RWNN variants, we perform 8 predictions per vertex during training, and average 32 predictions per vertex at test-time.

Results The results are in Table 5. All random walk based methods consistently perform better than GCN, SAGE, and GAT, and RWNN-transformer achieves the lowest test MSE. This supports our hypothesis that RWNNs can, in general, be resistant to over-smoothing and over-squashing (for the latter, if the walk is long), but also shows that proper design choices are important to obtain the best performance. For example, CRaWl and CRaWl* are limited by their window size of information processing (see Appendix A.10), and RWNN-MLP is bottlenecked by the fixed-sized hidden feature. While WalkLM-transformer performs on par with RWNN-transformer on Clique, its error is higher than RWNN-MLP on Barbell, showing the benefits of our design of walks and their records.
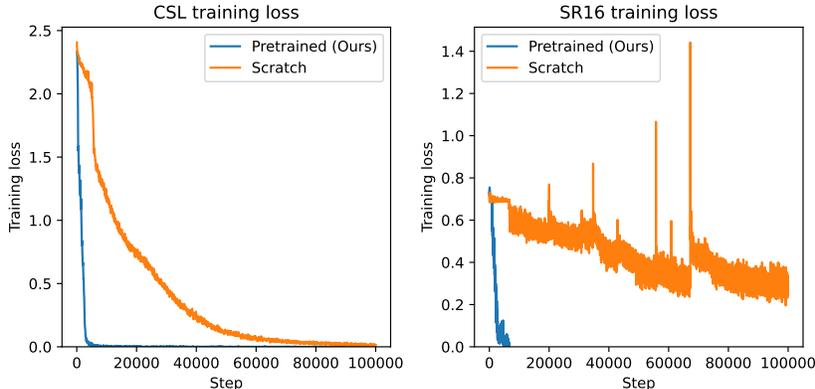
Figure 14: The impact of language pre-training on graph isomorphism learning (Section 5.2).

Table 6: Fine-tuning for arXiv, extending Table 4 (Section 5.3). † uses validation labels as in Table 4.

Method	Training-free?	Accuracy
Llama3-8b zero-shot	○	51.31%
Llama3-8b one-shot	○	52.81%
Llama3-8b one-shot†	○	52.82%
Llama3-70b zero-shot	○	65.30%
Llama3-70b one-shot	○	67.65%
RWNN-Llama3-8b (Ours)	○	71.06%
RWNN-Llama3-8b (Ours)†	○	73.17%
RWNN-Llama3-70b (Ours)†	○	74.85%
RWNN-Llama3-8b (Ours), fine-tuned	×	**74.95%**

Table 7: Transductive classification test accuracy on homophilic (Cora, Citeseer) and heterophilic (Amazon Ratings) datasets. The baselines scores are from Sato (2024); Zhao et al. (2023); Liu et al. (2024); Chen et al. (2024a;b); Platonov et al. (2023b). † denotes using validation labels as in Table 4.

Method	Training-free?	Cora 20-shot	Cora	Citeseer	Amazon Ratings
Training-free GNN	○	60.00%	-	-	-
GCN	×	81.40%	89.13%	74.92%	48.70%
GAT	×	80.80%	89.68%	75.39%	49.09%
GraphText	○	68.30%	67.77%	68.98%	-
LLaGA	○	-	59.59%	-	-
GraphText	×	-	87.11%	74.77%	-
LLaGA	×	-	88.86%	-	28.20%
OFA	×	75.90%	-	-	51.44%
RWNN-Llama3-8b (Ours)	○	72.29%	85.42%	79.62%	42.71%
RWNN-Llama3-8b (Ours)†	○	79.84%	87.08%	80.41%	38.66%
RWNN-Llama3-8b (Ours), fine-tuned	×	79.45%	86.72%	80.72%	55.76%

A.5 SUPPLEMENTARY MATERIALS FOR GRAPH ISOMORPHISM LEARNING (SECTION 5.2)

The impact of language pre-training In Section 5.2, we have used language pre-trained DeBERTa for graph isomorphism learning, even though the records of random walks do not resemble natural language (Appendix A.2 and A.3). To test if language pre-training is useful, we additionally trained RWNN-DeBERTa on CSL and SR16 datasets using the same configurations to our reported models but from random initialization. The training curves are shown in Figure 14. We find that language pre-training has significant benefits over random initialization for isomorphism learning tasks, as randomly initialized models converge very slowly for CSL, and fails to converge for SR16.

We conjecture that certain representations or computational circuits acquired in language pre-training are useful for processing text records of walks to recognize graphs. For example, some circuits specialized in recognizing certain syntactic structures in natural language could be repurposed during fine-tuning to detect skip links of CSL graphs from text records of walks. Our results are also consistent with prior observations on cross-model transfer of pre-trained language models to different domains such as image recognition (Lu et al., 2022; Rothermel et al., 2021).

Table 8: Peptides-func graph classification. The baseline scores are from Tönshoff et al. (2024).

Method	Test AP
GCN	0.6860
GINE	0.6621
GatedGCN	0.6765
Transformer	0.6326
SAN	0.6439
CRaWl	0.7074
RWNN-DeBERTa (Ours)	**0.7123 ± 0.0016**

```
Record  1[O,sp2]=π2[C,d:3,sp2,r]-3[C,cw,d:4,H:1,sp3,r]-4[N,d:3,H:1,sp2,r]-
        π5[C,d:3,sp2,r]=π6[O,sp2]=π5-π4-3-2=π1=π2-3-7[C,d:4,H:2,sp3,r]-8[S,d:2,sp3,r]-
        9[S,d:2,sp3,r]-10[C,d:4,H:2,sp3,r]-11[C,ccw,d:4,H:1,sp3,r]-12[N,d:3,H:1,sp2,r]-
        π13[C,d:3,sp2,r]=π14[O,sp2]=π13-π12-11-15[C,d:3,sp2,r]-π16[N,d:3,H:1,sp2,r]-
        17[C,cw,d:4,H:1,sp3,r]-18[C,d:3,sp2,r]=π19[O,sp2]=π18-17-20[C,d:4,H:2,sp3]-
        21[C,d:4,H:2,sp3]-22[C,d:3,sp2]=π23[O,sp2]=π22-π24[O,d:2,H:1,sp2]-π22=π23=
        π22-π24-π22=π23=π22-21-20-17-16-π15=π25[O,sp2]=π15-π16-17-20-21-22-π24-π22=
        π23=π22-21-20-17-16-π15=π25=π15-π16-17-20-21-22-π24-π22=π23=π22-π24-π22-21-
        20-17-18=π19=π18-π26[N,d:3,H:1,sp2,r]-27[C,ccw,d:4,H:1,sp3,r]-28[C,d:3,sp2]=
        π29[O,sp2]=π28-π30[N,d:3,H:1,sp2]-31[C,ccw,d:4,H:1,sp3]-32[C,d:4,H:1,sp3]-
        33[C,d:4,H:3,sp3]-32-34[C,d:4,H:3,sp3]-32-33-32-34-32-33-34-32-33-32-31-30-
        π28=π29=π28-π30-31-32-34-32-31-30-π28=π29=π28-π30-31-32-33-32-34-32-33-32-34-
        32-31-30-π28=π29=π28-π30-31-35[C,d:3,sp2]=36[O,sp2]=π35-π37[N,d:3,H:1,sp2]-
        38[C,cw,d:4,H:1,sp3]-39[C,d:3,sp2]-π40[N,d:3,H:1,sp2]-41[C,cw,d:4,H:1,sp3]-
        42[C,d:4,H:2,sp3]-43[C,d:3,sp2,*,r]*π44[C,d:3,H:1,sp2,*,r]*π45[N,d:3,H:1,sp2,*
        ,r]*π46[C,d:3,H:1,sp2,*,r]*π47[N,d:2,sp2,*,r]#*π43*π43*π44*π45*π46*π47*π43-42-41-
        40-π39=π48[O,sp2]=π39-38-37-π35=π36=π35-π37-38-49[C,d:4,H:2,sp3]-50[C
```

Figure 15: An example text record for Peptides-func graph classification.

A.6   SUPPLEMENTARY MATERIALS FOR TRANSDUCTIVE CLASSIFICATION (SECTION 5.3)

Fine-tuning   In Section 5.3, our RWNNs are training-free. We demonstrate fine-tuning the Llama 3 8b reader NN using the training labels with quantized low-rank adaptation (QLoRA) (Hu et al., 2022; Dettmers et al., 2023). The result is in Table 6, showing a promising performance.

Additional real-world transductive classification   We provide additional experiments on real-world transductive classification datasets Cora, Citeseer, and Amazon Ratings. Our RWNNs are designed similarly to the ones used for arXiv. We compile the baseline scores from Sato (2024); Zhao et al. (2023); Liu et al. (2024); Chen et al. (2024a;b); Platonov et al. (2023b), which includes MPNNs and language models on graphs. The results are in Table 7. Llama 3 based RWNN is competitive among training-free methods, and performs reasonably well when fine-tuned with QLoRA. Especially, on Amazon Ratings which is heterophilic (Platonov et al., 2023b), our fine-tuned RWNN-Llama3-8b achieves the highest accuracy (55.76%). This supports our results in Section 3.2 that RWNNs avoid over-smoothing, as avoiding it is known to be important for handling heterophily (Yan et al., 2022).

A.7   SUPPLEMENTARY TASKS

We provide supplementary experimental results on real-world graph classification, substructure counting, and link prediction tasks we could not include in the main text due to space constraints.

Real-world graph classification   We conduct a preliminary demonstration of RWNN on real-world graph classification, using the Peptides-func dataset (Dwivedi et al., 2022) used in Tönshoff et al. (2023). Our model is a pre-trained DeBERTa-base (identical to Section 5.2) fine-tuned on text records of non-backtracking MDLR walks with anonymization and named neighbor recording. The recording function is designed to properly incorporate the vertex and edge attributes provided in the dataset, including the atom and bond types. An example text is provided in Figure 15.

In Table 8, we report the test average precision (AP) at best validation accuracy, with 40 random predicted logits averaged at test time. We report the mean and standard deviation for five repeated tests. The baseline scores, including CRaWl, are from Tönshoff et al. (2023); Tönshoff et al. (2024). We see that RWNN-DeBERTa, which has been successful in graph isomorphism learning (Section 5.2), also shows a promising result in real-world protein graph classification, despite its simplicity of recording random walks in plain text and processing them with a fine-tuned language model.

Table 9: Substructure (8-cycle) counting. We report normalized test mean absolute error (MAE) at best validation. We trained MP, Subgraph GNN, Local 2-GNN, and Local 2-FGNN using configurations in Zhang et al. (2024), and trained MP-RNI, MP-Dropout, AgentNet, and CRaWl as in Table 3.

Method	Graph-level	Vertex-level
MP	0.0917	0.1156
Subgraph GNN	0.0515	0.0715
Local 2-GNN	0.0433	0.0478
Local 2-FGNN	0.0422	0.0438
Local 2-FGNN (large)	0.0932	0.1121
MP-RNI	0.3610	0.3650
MP-Dropout	0.1359	0.1393
AgentNet	0.1107	N/A
CRaWl	0.0526	0.0725
RWNN-DeBERTa (Ours)	0.0459	0.0465

Table 10: Real-world link prediction. The baseline scores are from Table 5 of Cai et al. (2021) and we reproduced LGLP using the official code. We report the aggregated test AP for 3 randomized runs.

Method	YST	KHN	ADV
Katz	81.63 ± 0.41	83.04 ± 0.38	91.76 ± 0.15
PR	82.08 ± 0.46	87.18 ± 0.26	92.43 ± 0.17
SR	76.02 ± 0.49	75.87 ± 0.66	83.22 ± 0.20
node2vec	76.61 ± 0.94	80.60 ± 0.74	76.70 ± 0.82
SEAL	86.45 ± 0.25	90.37 ± 0.16	93.52 ± 0.13
LGLP	88.54 ± 0.77	90.80 ± 0.33	93.51 ± 0.32
RWNN-transformer (Ours)	88.13 ± 0.45	90.90 ± 0.35	93.70 ± 0.21

Substructure counting   To supplement Section 5.2, we demonstrate RWNNs on challenging tasks that require both expressive power and generalization to unseen graphs. We use a synthetic dataset of 5,000 random regular graphs from Chen et al. (2020) and consider the task of counting substructures. We choose 8-cycles since they require a particularly high expressive power (Zhang et al., 2024). We test both graph- and vertex-level counting (i.e., counting 8-cycles containing the queried vertex) following Zhang et al. (2024). We use the data splits from Chen et al. (2020); Zhao et al. (2022b); Zhang et al. (2024), while excluding disconnected graphs following our assumption in Section 2.1. Our RWNN uses the same design to Section 5.2, and we train them with L1 loss for ≤250k steps using batch size of 128 graphs for graph-level counting and 8 graphs for vertex-level. At test-time, we ensemble 64 and 32 predictions by averaging for graph- and vertex-level tasks, respectively.

The results are in Table 9, and overall in line with Section 5.2. While MPNNs show a low performance, variants with higher expressive powers such as local 2-GNN and local 2-FGNN achieve high performances. This is consistent with the observation of Zhang et al. (2024). On the other hand, MPNNs with stochastic symmetry-breaking often show learning difficulties and does not achieve good performance. AgentNet was not directly applicable for vertex-level counting since its vertex encoding depends on which vertices the agent visits, and a learning difficulty was observed for graph-level counting. Our approach based on DeBERTa demonstrates competitive performance, outperforming random walk baselines and approaching performances of local 2-GNN and 2-FGNN.

Link prediction   We demonstrate an application of RWNNs to real-world link prediction, based on Cai et al. (2021) where link prediction is cast as vertex classification on line graphs. We follow the experimental setup of Cai et al. (2021) and only change their 3-layer 32-dim DGCNN with an RWNN that uses a 3-layer 32-dim transformer encoder. As the task is vertex-level, we use periodic restarts with $k = 5$. The results in Table 10 shows that RWNN performs reasonably well.

A.8   SUPPLEMENTARY ANALYSIS (SECTION 5 AND APPENDIX A.7)

We provide supplementary analysis and discussion on the impact of cover times, test-time ensembling, and the effective walk lengths, which we could not include in the main text due to space constraints.

Table 11: The performance of RWNN-DeBERTa on SR16 graph separation (Table 3, Section 5.2) for different required cover times controlled by the use of named neighbor recording.

	Test accuracy	Cover time
w/o named neighbor recording	50%	$C_E(G) = 490.00$
w/ named neighbor recording	**100%**	$C_V(G) = \mathbf{48.25}$

Table 12: The performance of RWNN-Llama3-70b† on arXiv transductive classification (Table 4, Section 5.3) for different local cover times controlled by the restart probability α of the random walk.

	Test accuracy	Cover time $C_V(B_{r=1}(v))$	Unique vertices observed per walk
$\alpha = 0.3$	74.40%	161.2	**45.28**
$\alpha = 0.7$	**74.85%**	**96.84**	31.13

The impact of cover times of walks In Section 2, we use cover times as a key tool for the analysis and comparison of different random walk algorithms. In the analysis, cover times provide the worst-case upper bounds of walk length required to achieve universality, which is in line with the literature on the cover times of Markov chains (Feige, 1995). To demonstrate how the differences in cover times may translate to different model behaviors, we provide two examples where performing interventions that change cover times leads to substantial differences in task performances.

First, in SR16 graph separation (Section 5.2), we test the use of named neighbor recording, which changes the required cover time from vertex cover time $C_V(G)$ (when used) to edge cover time $C_E(G)$ (when not used). The result is in Table 11. We can see that not using named neighbor recording has a drastic effect on the required cover time (vertex cover time $48.25 \rightarrow$ edge cover time[6] 490.00), and the test performance deteriorates from perfect to random accuracy ($100\% \rightarrow 50\%$).

Second, in arXiv transductive classification (Section 5.3), due to the high homophily (Platonov et al., 2023a) we can assume that the cover time of the local neighborhoods, i.e., $B_r(v)$ for small r, has an important influence on performance. Thus, we set $r = 1$ and check if the walk strategy that lowers the cover time of $B_r(v)$ leads to a better performance. To test different local cover times, we use two choices of the restart probability $\alpha = 0.3$ and 0.7. Since we use named neighbor recording, we measure vertex cover times. Due to the large size of G, we estimate the cover times by fixing a random set of 100 test vertices and running 100 walks from each of them. The result is in Table 12. Increasing the restart probability ($0.3 \rightarrow 0.7$) simultaneously results in a lower cover time ($161.2 \rightarrow 96.84$) and a higher test accuracy ($74.40\% \rightarrow 74.85\%$), even though the average number of observed vertices decreases ($45.28 \rightarrow 31.13$). This implies that lower cover time is correlated with better performance, while the number of observed vertices itself is not directly correlated with better performance.

While the above shows cases where interventions in cover times lead to different task performances, in general, there are factors that make it not always trivial to equate task performances and cover times. This is because a substantial amount of useful information in practice could be recovered from possibly non-covering walks. We discuss this aspect in depth in Appendix A.11.

The impact of test-time ensembling For each test input to an RWNN X_θ, we often sample several randomized predictions $\hat{\mathbf{y}} \sim X_\theta(\cdot)$ and ensemble them by averaging (or voting, for Llama 3 models where we cannot directly obtain continuous logits). As discussed in Section 2, this can be viewed as MC estimation of the underlying invariant mean predictor $(\cdot) \mapsto \mathbb{E}[X_\theta(\cdot)]$ (or the invariant limiting classifier in the case of voting (Cotta et al., 2023)). From another perspective, we can also interpret this as a type of test-time scaling with repeated sampling, which has been shown beneficial in language models (Brown et al., 2024) and randomized algorithms (called amplification (Cotta et al., 2023; Sieradzki et al., 2022)). To see if similar perspectives can be held for RWNNs, we measure the model performances with increased numbers of samples for averaging or voting. The results are in Table 13. We see that increasing sample size consistently leads to better performances and reduced variances overall. The point of plateau seems to differ per application and model architecture, and we leave obtaining a better understanding of the scaling behavior for future work.

---

[6]As in Section 5.1, while the strict definition of edge cover requires traversing each edge in both directions, our measurements only require traversing one direction, which suffices for the universality of RWNNs.

Table 13: Test performances of RWNN-Llama3-70b† (Table 4, Section 5.3) and RWNN-DeBERTa (Table 8, Appendix A.7) for different numbers of randomized predictions for ensembling by voting and averaging, respectively. For *, we were unable to run repeated experiments due to sampling costs.

# samples	arXiv test accuracy
1	74.45 ± 0.049%
3	74.75 ± 0.035%
5	74.83 ± 0.028%
7	74.86% ± N/A*
10	**74.87% ± N/A***

# samples	Peptides-func test AP
1	0.5300 ± 0.0092
10	0.6937 ± 0.0039
20	0.7056 ± 0.0026
40	0.7123 ± 0.0016
60	0.7145 ± 0.0015
80	0.7155 ± 0.0014
100	0.7161 ± 0.0009
160	0.7176 ± 0.0013
320	**0.7177 ± N/A***

Table 14: Effective random walk lengths for the experiments in Sections 5.2 and 5.3.

Dataset	Length
CSL (Section 5.2)	214.03 ± 2.48
SR16 (Section 5.2)	222.00 ± 0.00
SR25 (Section 5.2)	129.54 ± 2.26
arXiv (Section 5.3)	194.52 ± 10.59

Effective walk lengths For Sections 5.2 and 5.3, the effective lengths of walks are affected by the language model tokenizers since text records exceeding the maximum number of tokens are truncated. The maximum number of tokens is a hyperparameter which we determine from memory constraints; for the DeBERTa tokenizer used in Section 5.2, we set it to 512, and for the Llama 3 tokenizer used in Section 5.3, we set it to 4,096. The effective lengths of random walks for each dataset are measured and provided in Table 14. The effective length is shorter (129.54 on average) for SR25 graphs compared to CSL and SR16 (214.03 and 222.00 on average, respectively) although using the same tokenizer configurations, which is because the higher average degrees of SR25 graphs (Figure 6) compared to CSL and SR16 graphs (Figures 4 and 5, respectively) has led to a higher number of text characters invested in recording named neighbors ("#" symbols in the figures).

A.9 PROOFS

We recall that an isomorphism between two graphs G and H is a bijection $\pi : V(G) \to V(H)$ such that any two vertices u and v are adjacent in G if and only if $\pi(u)$ and $\pi(v)$ are adjacent in H. If an isomorphism π exists between G and H, the graphs are isomorphic and written $G \simeq H$ or $G \overset{\pi}{\simeq} H$. By $X \overset{d}{=} Y$ we denote the equality of two random variables X and Y in distributions.

In the proofs, we write by $\mathrm{id}(\cdot)$ the namespace obtained by anonymization of a walk $v_0 \to \cdots \to v_l$ (Section 2) that maps each vertex v_t to its unique integer name $\mathrm{id}(v_t)$ starting from $\mathrm{id}(v_0) = 1$.

Let us define some notations related to cover times. These will be used from Appendix A.9.5. We denote by $H(u, v)$ the expected number of steps a random walk starting from u takes until reaching v. This is called the hitting time between u and v. For a graph G, let $C_V(G, u)$ be the expected number of steps a random walk starting from u takes until visiting every vertices of G. The vertex cover time $C_V(G)$, or the cover time, is this quantity given by the worst possible u:

$$C_V(G) := \max_{u \in V(G)} C_V(G, u). \tag{16}$$

Likewise, let $C_E(G, u)$ be the expected number of steps a random walk starting from u takes until traversing every edge of G. The edge cover time $C_E(G)$ is given by the worst possible u:

$$C_E(G) := \max_{u \in V(G)} C_E(G, u). \tag{17}$$

For local balls $B_r(u)$, we always set the starting vertex of the random walk to u (Section 2.2). Thus, we define their cover times as follows:

$$C_V(B_r(u)) := C_V(B_r(u), u), \tag{18}$$
$$C_E(B_r(u)) := C_E(B_r(u), u). \tag{19}$$

A.9.1 PROOF OF PROPOSITION 2.1 (SECTION 2)

Proposition 2.1. *$X_\theta(\cdot)$ is invariant in probability, if its random walk algorithm is invariant in probability and its recording function is invariant.*

Proof. We recall the probabilistic invariance of random walk algorithm in Equation 3:

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \quad \forall G \overset{\pi}{\simeq} H, \tag{20}$$

where $v_{[\cdot]}$ is a random walk on G and $u_{[\cdot]}$ is a random walk on H. We further recall the invariance of recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ in Equation 7:

$$q(v_0 \to \cdots \to v_l, G) = q(\pi(v_0) \to \cdots \to \pi(v_l), H), \quad \forall G \overset{\pi}{\simeq} H, \tag{21}$$

for any given random walk $v_{[\cdot]}$ on G. Combining Equation 20 and equation 21, we have the following:

$$q(v_0 \to \cdots \to v_l, G) \overset{d}{=} q(u_0 \to \cdots \to u_l, H), \quad \forall G \simeq H. \tag{22}$$

Then, since the reader neural network f_θ is a deterministic map, we have:

$$f_\theta(q(v_0 \to \cdots \to v_l, G)) \overset{d}{=} f_\theta(q(u_0 \to \cdots \to u_l, H)), \quad \forall G \simeq H, \tag{23}$$

which leads to:

$$X_\theta(G) \overset{d}{=} X_\theta(H), \quad \forall G \simeq H. \tag{24}$$

This shows Equation 2 and completes the proof. $\qquad\square$

A.9.2 PROOF OF PROPOSITION 2.2 (SECTION 2)

We first show a useful lemma.

Lemma A.2. *The random walk in Equation 4 is invariant in probability, if the probability distributions of the starting vertex v_0 and each transition $v_{t-1} \to v_t$ are invariant.*

37

Proof. We prove by induction. Let us write probabilistic invariance of the starting vertex v_0 as:

$$\pi(v_0) \overset{d}{=} u_0, \quad \forall G \overset{\pi}{\simeq} H, \tag{25}$$

and probabilistic invariance of each transition $v_{t-1} \to v_t$ as follows, by fixing v_{t-1} and u_{t-1}:

$$\pi(v_{t-1}) \to \pi(v_t) \overset{d}{=} u_{t-1} \to u_t, \quad \forall G \overset{\pi}{\simeq} H, v_{t-1} \in V(G), u_{t-1} := \pi(v_{t-1}). \tag{26}$$

Let us assume the following for some $t \geq 1$:

$$\pi(v_0) \to \cdots \to \pi(v_{t-1}) \overset{d}{=} u_0 \to \cdots \to u_{t-1}, \quad G \overset{\pi}{\simeq} H. \tag{27}$$

Then, from the chain rule of joint distributions, the first-order Markov property of random walk in Equation 4, and probabilistic invariance in Equation 26, we obtain the following:

$$\pi(v_0) \to \cdots \to \pi(v_t) \overset{d}{=} u_0 \to \cdots \to u_t, \quad G \overset{\pi}{\simeq} H. \tag{28}$$

By induction from the initial condition $\pi(v_0) \overset{d}{=} u_0$ in Equation 25, the following holds $\forall l > 0$:

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \quad \forall G \overset{\pi}{\simeq} H. \tag{29}$$

This shows Equation 3 and completes the proof. $\qquad\square$

We now prove Proposition 2.2.

Proposition 2.2. *The random walk in Equation 4 is invariant in probability if its conductance $c_{[\cdot]}(\cdot)$ is invariant:*

$$c_G(u, v) = c_H(\pi(u), \pi(v)), \quad \forall G \overset{\pi}{\simeq} H. \tag{30}$$

It includes constant conductance, and any choice that only uses degrees of endpoints $\deg(u)$, $\deg(v)$.

Proof. We recall Equation 4 for a given conductance function $c_G : E(G) \to \mathbb{R}_+$:

$$\mathrm{Prob}[v_t = x | v_{t-1} = u] := \frac{c_G(u, x)}{\sum_{y \in N(u)} c_G(u, y)}. \tag{31}$$

From Lemma A.2, it is sufficient to show the probabilistic invariance of each transition $v_{t-1} \to v_t$ as we sample v_0 from invariant distribution $\mathrm{Uniform}(V(G))$ (Algorithm 1). We rewrite Equation 26 as:

$$\mathrm{Prob}[v_t = x | v_{t-1} = u] = \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(u)], \quad \forall G \overset{\pi}{\simeq} H. \tag{32}$$

If the conductance function $c_{[\cdot]}(\cdot)$ is invariant (Equation 30), we can show Equation 32 by:

$$\begin{aligned}
\mathrm{Prob}[v_t = x | v_{t-1} = u] &:= \frac{c_G(u, x)}{\sum_{y \in N(u)} c_G(u, y)}, \\
&= \frac{c_H(\pi(u), \pi(x))}{\sum_{y \in N(u)} c_H(\pi(u), \pi(y))}, \\
&= \frac{c_H(\pi(u), \pi(x))}{\sum_{\pi(y) \in N(\pi(u))} c_H(\pi(u), \pi(y))}, \\
&= \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(u)], \quad \forall G \overset{\pi}{\simeq} H.
\end{aligned} \tag{33}$$

In the third equality, we used the fact that isomorphism $\pi(\cdot)$ preserves adjacency. It is clear that any conductance function $c_{[\cdot]}(\cdot)$ with a constant conductance such as $c_G(\cdot) = 1$ is invariant. Furthermore, any conductance function that only uses degrees of endpoints $\deg(u)$, $\deg(v)$ is invariant as isomorphism $\pi(\cdot)$ preserves the degree of each vertex. This completes the proof. $\qquad\square$

A.9.3 PROOF OF PROPOSITION A.1 (APPENDIX A.1)

We extend the proof of Proposition 2.2 given in Appendix A.9.2 to second-order random walks discussed in Appendix A.1. We first show a useful lemma:

Lemma A.3. *A second-order random walk algorithm is invariant in probability, if the probability distributions of the starting vertex v_0, the first transition $v_0 \to v_1$, and each transition $(v_{t-2}, v_{t-1}) \to v_t$ for $t > 1$ are invariant.*

Proof. We prove by induction. The probabilistic invariance of starting vertex v_0 and first transition $v_0 \to v_1$ are:

$$\pi(v_0) \overset{d}{=} u_0, \quad \forall G \overset{\pi}{\simeq} H, \tag{34}$$

$$\pi(v_0) \to \pi(v_1) \overset{d}{=} u_0 \to u_1, \quad \forall G \overset{\pi}{\simeq} H, v_0 \in V(G), u_0 := \pi(v_0), \tag{35}$$

and probabilistic invariance of each transition $(v_{t-2}, v_{t-1}) \to v_t$ for $t > 1$ can be written as follows by fixing (v_{t-2}, v_{t-1}) and (u_{t-2}, u_{t-1}):

$$\pi(v_{t-2}) \to \pi(v_{t-1}) \to \pi(v_t) \overset{d}{=} u_{t-2} \to u_{t-1} \to u_t, \quad \forall G \overset{\pi}{\simeq} H,$$
$$(v_{t-2}, v_{t-1}) \in E(G),$$
$$(u_{t-2}, u_{t-1}) := (\pi(v_{t-2}), \pi(v_{t-1})). \tag{36}$$

Let us assume the following for some $t \geq 2$:

$$\pi(v_0) \to \cdots \to \pi(v_{t-2}) \to \pi(v_{t-1}) \overset{d}{=} u_0 \to \cdots \to u_{t-2} \to u_{t-1}, \quad G \overset{\pi}{\simeq} H. \tag{37}$$

Then, from the chain rule of joint distributions, the second-order Markov property of second-order random walks, and probabilistic invariance in Equation 36, we obtain the following:

$$\pi(v_0) \to \cdots \to \pi(v_{t-1}) \to \pi(v_t) \overset{d}{=} u_0 \to \cdots \to u_{t-1} \to u_t, \quad G \overset{\pi}{\simeq} H. \tag{38}$$

By induction from the initial condition $\pi(v_0) \to \pi(v_1) \overset{d}{=} u_0 \to u_1$ given from Equations 34 and 35, the following holds $\forall l > 0$:

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \quad \forall G \overset{\pi}{\simeq} H. \tag{39}$$

This shows Equation 3 and completes the proof. $\qquad\square$

We now prove Proposition A.1.

Proposition A.1. *The non-backtracking random walk in Equation 13 and the node2vec random walk in Equation 14 are invariant in probability, if their underlying first-order random walk algorithm is invariant in probability.*

Proof. We assume that the first transition $v_0 \to v_1$ is given by the underlying first-order random walk algorithm. This is true for non-backtracking since there is no v_{t-2} to avoid, and true for the official implementation of node2vec (Grover & Leskovec, 2016). Then from Lemma A.3, it is sufficient to show probabilistic invariance of each transition $(v_{t-2}, v_{t-1}) \to v_t$ for $t > 1$ as we sample v_0 from the invariant distribution $\text{Uniform}(V(G))$ (Algorithm 1) and the first transition $v_0 \to v_1$ is given by the first-order random walk which is assumed to be invariant in probability. Rewrite Equation 36 as:

$$\text{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] = \text{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \quad \forall G \overset{\pi}{\simeq} H. \tag{40}$$

For non-backtracking, we first handle the case where $i \to j$ reaches a dangling vertex j which has i as its only neighbor. In this case the walk begrudgingly backtracks $i \to j \to i$ (Appendix A.1). Since isomorphism $\pi(\cdot)$ preserves adjacency, $\pi(i) \to \pi(j)$ also reaches a dangling vertex $\pi(j)$ and must begrudgingly backtrack $\pi(i) \to \pi(j) \to \pi(i)$. By interpreting the distributions on v_t and u_t as one-hot at i and $\pi(i)$, respectively, we can see that Equation 40 holds. If $i \to j$ does not reach a

dangling vertex, the walk $j \to x$ follows the invariant probability of the underlying first-order random walk $\mathrm{Prob}[v_{t+1} = x | v_t = j]$ renormalized over $N(j) \setminus \{i\}$. Then we can show Equation 40 by:

$$\mathrm{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i]$$

$$:= \begin{cases} \dfrac{\mathrm{Prob}[v_t = x | v_{t-1} = j]}{\sum_{y \in N(j) \setminus \{i\}} \mathrm{Prob}[v_t = y | v_{t-1} = j]} & \text{for } x \neq i, \\ \\ 0 & \text{for } x = i, \end{cases}$$

$$= \begin{cases} \dfrac{\mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j)]}{\sum_{\pi(y) \in N(\pi(j)) \setminus \{\pi(i)\}} \mathrm{Prob}[u_t = \pi(y) | u_{t-1} = \pi(j)]} & \text{for } \pi(x) \neq \pi(i), \\ \\ 0 & \text{for } \pi(x) = \pi(i), \end{cases}$$

$$= \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \quad \forall G \overset{\pi}{\simeq} H. \tag{41}$$

In the second equality, we have used the fact that isomorphism $\pi(\cdot)$ is a bijection and preserves adjacency, and the assumption that the first-order random walk algorithm is invariant in probability. For node2vec walk, we first show the invariance of the weighting term $\alpha(i, x)$ in Equation 15 using the fact that isomorphism preserves shortest path distances between vertices $d(i, x) = d(\pi(i), \pi(x))$:

$$\alpha(i, x) := \begin{cases} 1/p & \text{for } d(i, x) = 0, \\ 1 & \text{for } d(i, x) = 1, \\ 1/q & \text{for } d(i, x) = 2. \end{cases}$$

$$= \begin{cases} 1/p & \text{for } d(\pi(i), \pi(x)) = 0, \\ 1 & \text{for } d(\pi(i), \pi(x)) = 1, \\ 1/q & \text{for } d(\pi(i), \pi(x)) = 2. \end{cases}$$

$$= \alpha(\pi(i), \pi(x)). \tag{42}$$

Then we can show Equation 40 by:

$$\mathrm{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] := \frac{\alpha(i, x) \, \mathrm{Prob}[v_t = x | v_{t-1} = j]}{\sum_{y \in N(j)} \alpha(i, y) \, \mathrm{Prob}[v_t = y | v_{t-1} = j]},$$

$$= \frac{\alpha(\pi(i), \pi(x)) \, \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j)]}{\sum_{\pi(y) \in N(\pi(j))} \alpha(\pi(i), \pi(y)) \, \mathrm{Prob}[u_t = \pi(y) | u_{t-1} = \pi(j)]},$$

$$= \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \quad \forall G \overset{\pi}{\simeq} H. \tag{43}$$

In the second equality, we have used the fact that isomorphism $\pi(\cdot)$ preserves adjacency, and the assumption that the first-order walk algorithm is invariant in probability. This completes the proof. $\square$

A.9.4 PROOF OF PROPOSITION 2.3 (SECTION 2)

Proposition 2.3. *A recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ that uses anonymization, optionally with named neighbors, is invariant.*

Proof. Given a walk $v_0 \to \cdots \to v_l$ on G, we can write the anonymized namespace $\mathrm{id}(\cdot)$ as follows:

$$\mathrm{id}(v_t) = 1 + \arg\min_i [v_i = v_t]. \tag{44}$$

Consider any walk $\pi(v_0) \to \cdots \to \pi(v_l)$ on some H which is isomorphic to G via π. Let us denote the anonymization of this walk as $\mathrm{id}'(\cdot)$. Then we have:

$$\mathrm{id}'(\pi(v_t)) = 1 + \arg\min_i [\pi(v_i) = \pi(v_t)],$$

$$= 1 + \arg\min_i [v_i = v_t],$$

$$= \mathrm{id}(v_t), \quad \forall G \overset{\pi}{\simeq} H. \tag{45}$$

The second equality is due to the fact that π is a bijection. This completes the proof for anonymization. For the named neighbors, we prove by induction. Let us fix some $t \geq 1$ and assume:

$$q(v_0 \to \cdots \to v_{t-1}, G) = q(\pi(v_0) \to \cdots \to \pi(v_{t-1}), H), \quad \forall G \overset{\pi}{\simeq} H. \tag{46}$$

Let $T \subseteq E(G)$ be the set of edges recorded by $\mathbf{z} := q(v_0 \to \cdots \to v_{t-1}, G)$, and $T' \subseteq E(H)$ be the set of edges recorded by $\mathbf{z}' := q(\pi(v_0) \to \cdots \to \pi(v_{t-1}), H)$. Then, T and T' are isomorphic via π:

$$T' = \{(\pi(u), \pi(v)) \forall (u, v) \in T\}. \tag{47}$$

The equality is shown as follows. ($\supseteq$) Any $(u, v) \in T$ is recorded in $\mathbf{z}$ as $(\mathrm{id}(u), \mathrm{id}(v))$. From Equations 46 and 45, we have $\mathbf{z} = \mathbf{z}'$ and $(\mathrm{id}(u), \mathrm{id}(v)) = (\mathrm{id}(\pi(u)), \mathrm{id}(\pi(v)))$. Thus, $(\pi(u), \pi(v))$ is recorded in $\mathbf{z}'$ as $(\mathrm{id}(\pi(u)), \mathrm{id}(\pi(v)))$, i.e., $(\pi(u), \pi(v)) \in T'$. ($\subseteq$) This follows from symmetry. Now, we choose some $v_t \in N(v_{t-1})$ and consider the set $F \subseteq E(G)$ of all unrecorded edges from v_t:

$$F := \{(v_t, u) : u \in N(v_t)\} \setminus T. \tag{48}$$

Then, the set D of unrecorded neighbors of v_t is given as:

$$D := \{u : (v_t, u) \in F\}. \tag{49}$$

Since $\pi(\cdot)$ preserves adjacency, the set $F' \subseteq E(H)$ of all unrecorded edges from $\pi(v_t)$ is given by:

$$\begin{aligned} F' &:= \{(\pi(v_t), \pi(u)) : \pi(u) \in N(\pi(v_t))\} \setminus T', \\ &= \{(\pi(v_t), \pi(u)) : u \in N(v_t)\} \setminus \{(\pi(u'), \pi(v')) \forall (u', v') \in T\}, \\ &= \{(\pi(v_t), \pi(u)) : u \in N(v_t), (v_t, u) \notin T\}, \\ &= \{(\pi(v_t), \pi(u)) : (v_t, u) \in F\}, \end{aligned} \tag{50}$$

and consequently:

$$\begin{aligned} D' &:= \{\pi(u) : (\pi(v_t), \pi(u)) \in F'\}, \\ &= \{\pi(u) : u \in D\}. \end{aligned} \tag{51}$$

While D and D' may contain vertices not yet visited by the walks and hence not named by $\mathrm{id}(\cdot)$, what we record are named neighbors. Let S be the set of all named vertices in G. It is clear that the set S' of named vertices in H is given by $S' = \{\pi(v) : v \in S\}$. Then, the named neighbors in G to be newly recorded for $v_{t-1} \to v_t$ is given by $U = D \cap S$, and for $\pi(v_{t-1}) \to \pi(v_t)$ in H the set is given by $U' = D' \cap S'$. Since $\pi(\cdot)$ is a bijection, we can see that $U' = \{\pi(u) : u \in U\}$. From the invariance of anonymization in Equation 45, U and U' are named identically $\{\mathrm{id}(u) : u \in U\} = \{\mathrm{id}(\pi(u)) : \pi(u) \in U'\}$, and therefore will be recorded identically. As a result, the information to be added to the records at time t are identical, and we have:

$$q(v_0 \to \cdots \to v_t, G) = q(\pi(v_0) \to \cdots \to \pi(v_t), H), \quad \forall G \overset{\pi}{\simeq} H. \tag{52}$$

Then, by induction from the initial condition:

$$q(v_0, G) = q(\pi(v_0), H) = \mathrm{id}(v_0) = \mathrm{id}(\pi(v_0)) = 1, \tag{53}$$

the recording function using anonymization and named neighbors is invariant. $\qquad \square$

A.9.5 PROOF OF THEOREM 2.4 (SECTION 2.2)

Theorem 2.4. *For a uniform random walk on an infinite graph G starting at v, the vertex and edge cover times of the finite local ball $B_r(v)$ are not always finitely bounded.*

Proof. We revisit the known fact that hitting times on the infinite line $\mathbb{Z}$ is infinite (Dumitriu et al., 2003; Janson & Peres, 2012; carnage, 2012; McNew, 2013). Consider a local ball $B_1(0)$ of radius 1 from the origin 0 such that $V(B_1(0)) = \{-1, 0, 1\}$. Let us assume that the expected number of steps for a random walk starting at 0 to visit 1 for the first time is finite, and denote it by T. In the first step of the walk, we have to go either to -1 or 1. If we go to -1, we have to get back to 0 and then to 1 which would take $2T$ in expectation (by translation symmetry). This leads to $T = 1/2 \cdot 1 + 1/2 \cdot (1 + 2T) = 1 + T$, which is a contradiction. Since visiting all vertices or traversing all edges in in $B_1(0)$ clearly involves visiting 1, the vertex and edge cover times cannot be finitely bounded. $\qquad \square$

A.9.6   Proof of Theorem 2.5 (Section 2.2)

Let us recall that our graph G is locally finite (Section 2.2), and denote its maximum degree as Δ. We further denote by $d(u, v)$ the shortest path distance between u and v. We show some useful lemmas. We first show that $H(u, x)$, the expected number of steps of a random walk starting at u takes until reaching x, is finite for any u, x in $B_r(v)$ for any nonzero restart probability α or restart period $k \geq r$.

Lemma A.4. *For any u and x in $B_r(v)$, $H(u, x)$ is bounded by the following:*

$$H(u, x) \leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^r + \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r}, \tag{54}$$

if the random walk restarts at v with any probability $\alpha \in (0, 1)$, and:

$$H(u, x) \leq k + k\Delta^r, \tag{55}$$

if the random walk restarts at v with any period $k \geq r$.

Proof. We first prove for random restarts. The proof is inspired by Theorem 1.1 of McNew (2013) and Lemma 2 of Zuckerman (1991). We clarify some measure-theoretic details. Let W be the set of all (infinite) random walks on G. We denote by P the probability measure defined on the space W, equipped with its cylinder σ-algebra, by the random walk restarting at v with probability $\alpha \in (0, 1)$. For a vertex x in G, we define the hitting time function $|\cdot|_x : W \to \mathbb{N} \cup \{\infty\}$ as follows:

$$|w|_x := \arg\min_i [(w)_i = x], \quad \forall w \in W. \tag{56}$$

Let $W(a)$ be the set of all random walks that start at a, $W(a, b)$ be the set of random walks that start at a and visit b, and $W(a, b^-)$ be the set of random walks that start at a and do not visit b. Then, the measurability of $|\cdot|_x$ follows from the measurability of the sets $W(a)$, $W(a, b)$, and $W(a, b^-)$, which we show as follows. $\forall a, b \in V(G)$, $W(a)$ is clearly measurable, $W(a, b)$ is measurable as it equals $\bigcup_{n \in \mathbb{N}} \{w \in W | (w)_1 = a, (w)_n = b\}$, and $W(a, b^-)$ is measurable as it is $W(a) \setminus W(a, b)$.

Consider $W(u, x^-)$, the set of random walks that start at u and never visit x. We start by showing that this set is of measure zero:

$$P(W(u, x^-)) = 0. \tag{57}$$

For this, we use the fact that $W(u, x^-) = W(u, v^-, x^-) \cup W(u, v, x^-)$, where $W(u, v^-, x^-)$ is the set of random walks starting at u that do not visit v nor x, and $W(u, v, x^-)$ is the set of walks starting at u that visit v and do not visit x. We have the following:

$$\begin{aligned} P(W(u, x^-)) &= P(W(u, v^-, x^-)) + P(W(u, v, x^-)), \\ &\leq P(W(u, v^-)) + P(W(u, v, x^-)). \end{aligned} \tag{58}$$

We show Equation 57 by showing $P(W(u, v^-)) = 0$ and $P(W(u, v, x^-)) = 0$ in Equation 58.

1. $(P(W(u, v^-)) = 0)$ Consider $W(u, v^-)$, the set of all random walks that start at u and never visit v. Since restart sends a walk to v, every walk in this set never restarts. The probability of a walk not restarting until step t is $(1-\alpha)^t$. Denote this probability by p_t, and let p be the probability of a random walk to never restart. Then $p_t \downarrow p = 0$ and $P(W(u, v^-)) \leq p = 0$.

2. $(P(W(u, v, x^-)) = 0)$ Assume $P(W(u, v, x^-)) > 0$. Then $P(W(v, x^-)) > 0$ since each walk step is independent. Let $W_N(v, x^-)$ be the set of walks that start at v and do not reach x within N restarts. Then we have $W_N(v, x^-) \downarrow W(v, x^-)$. If a walk restarts at v, the probability of reaching x before the next restart is at least the probability of exactly walking the shortest path from v to x, which is $\geq (\frac{1-\alpha}{\Delta})^{d(v,x)} \in (0, 1)$. Then $P(W_N(v, x^-)) \leq (1 - (\frac{1-\alpha}{\Delta})^{d(v,x)})^N \downarrow 0$, leading to $P(W(v, x^-)) = 0$. This is a contradiction, so we have $P(W(u, v, x^-)) = 0$.

We are now ready to bound $H(u, x)$. Using the hitting time function $|\cdot|_x$ in Equation 56, we have:

$$H(u, x) = \mathbb{E}\left[|w|_x | w \in W(u)\right]. \tag{59}$$

Since $W(u) = W(u, x) \cup W(u, x^-)$, and $P(W(u, x^-)) = 0$ from Equation 57, we have:

$$H(u, x) = \mathbb{E}[|w|_x | w \in W(u, x)],$$

$$= \int_{W(u,x)} |w|_x \, dP(w | w \in W(u, x)). \tag{60}$$

Each walk in $W(u, x)$ starts at u and, when it reaches x, it either has or has not visited v. We treat the two cases separately. Let $W(a, b, c)$ be the set of walks that start at a and reach c after b, and let $W(a, b^-, c)$ be the set of walks that start at a and reach c before b. Then we have:

$$H(u, x) = \int_{W(u,v,x) \cup W(u,v^-,x)} |w|_x \, dP(w | W(u, x)),$$

$$= \int_{W(u,v,x)} |w|_x \, dP(w | W(u, x)) + \int_{W(u,v^-,x)} |w|_x \, dP(w | W(u, x)). \tag{61}$$

Let $\hat{H}(a, b, c)$ (or $\hat{H}(a, b^-, c)$) be the expected number of steps for a walk starting at a to reach c after reaching b (or before b), given that it is in $W(a, b, c)$ (or in $W(a, b^-, c)$). If a walk from u reaches v before x, the expected steps is given by $\hat{H}(u, x^-, v) + H(v, x)$. If the walk reaches x before v, the expected steps is $\hat{H}(u, v^-, x)$. Then, if $W(u, v^-, x) \neq \emptyset$, we can write $H(u, x)$ as follows:

$$H(u, x) = \left[\hat{H}(u, x^-, v) + H(v, x) \right] P(W(u, v, x) | W(u, x))$$

$$+ \hat{H}(u, v^-, x) \, P(W(u, v^-, x) | W(u, x)). \tag{62}$$

On the other hand, if $W(u, v^-, x) = \emptyset$, we simply have:

$$H(u, x) = \hat{H}(u, x^-, v) + H(v, x). \tag{63}$$

We first consider $\hat{H}(u, x^-, v)$, the expected number of steps from u to reach v before x. We show:

$$\hat{H}(u, x^-, v) \leq \frac{1}{\alpha}. \tag{64}$$

To see this, note that $\hat{H}(u, x^-, v)$ is equivalent to the expectation $\mathbb{E}[T]$ of the number of steps T to reach v from u, on the graph G with the vertex x deleted and transition probabilities renormalized. If u is isolated on this modified graph, the only walk in $W(u, x^-, v)$ is the one that immediately restarts at v, giving $\mathbb{E}[T] = 1$. Otherwise, we have $\mathbb{E}[T] \leq 1/\alpha$ due to the following. Let T' be the number of steps until the first restart at v. Since restart can be treated as a Bernoulli trial with probability of success α, T' follows geometric distribution with expectation $1/\alpha$. Since it is possible for the walk to reach v before the first restart, we have $\mathbb{E}[T] \leq \mathbb{E}[T'] = 1/\alpha$, which gives Equation 64.

We now consider $H(v, x)$, the expectation $\mathbb{E}[T]$ of the number of steps T to reach x from v. Let T' be the steps until the walk restarts at v, then exactly walks the shortest path from v to x for the first time, and then restarts at v. Since T' walks until restart after walking the shortest path to x, and it is possible for a walk to reach x before walking the shortest path to it, we have $\mathbb{E}[T] \leq \mathbb{E}[T']$. Then, we split the walk of length T' into N trials, where each trial consists of restarting at v and walking until the next restart. A trial is successful if it immediately walks the shortest path from v to x. Then N is the number of trials until we succeed, and it follows geometric distribution with probability of success at least $(\frac{1-\alpha}{\Delta})^{d(v,x)}$ due to bounded degrees. Its expectation is then bounded as:

$$\mathbb{E}[N] \leq \left(\frac{\Delta}{1 - \alpha} \right)^{d(v,x)}. \tag{65}$$

Let S_i be the length of the i-th trial. Since each S_i is i.i.d. with finite mean $\mathbb{E}[S_i] = 1/\alpha$, and N is stopping time, we can apply Wald's identity (Hein) to compute the expectation of T':

$$\mathbb{E}[T'] = \mathbb{E}[S_1 + \cdots + S_N],$$

$$= \mathbb{E}[N] \mathbb{E}[S_1],$$

$$\leq \frac{1}{\alpha} \left(\frac{\Delta}{1 - \alpha} \right)^{d(v,x)}. \tag{66}$$

We remark that $H(v, x) \leq \mathbb{E}[T']$. Combining this result with Equation 64, we have:

$$\hat{H}(u, x^-, v) + H(v, x) \leq \frac{1}{\alpha} + \frac{1}{\alpha} \left(\frac{\Delta}{1 - \alpha} \right)^{d(v,x)} . \tag{67}$$

If $W(u, v^-, x) = \emptyset$, we have $H(u, x) = \hat{H}(u, x^-, v) + H(v, x)$ from Equation 63 and it is finitely bounded for any $\alpha \in (0, 1)$ by the above. If $W(u, v^-, x) \neq \emptyset$, Equations 62 and 67 lead to:

$$H(u, x) \leq \hat{H}(u, x^-, v) + H(v, x) + \hat{H}(u, v^-, x) \, P(W(u, v^-, x) | W(u, x)),$$

$$\leq \frac{1}{\alpha} + \frac{1}{\alpha} \left(\frac{\Delta}{1 - \alpha} \right)^{d(v,x)} + \hat{H}(u, v^-, x) \, P(W(u, v^-, x) | W(u, x)), \tag{68}$$

and it suffices to bound $\hat{H}(u, v^-, x) \, P(W(u, v^-, x) | W(u, x))$. We show the following:

$$\hat{H}(u, v^-, x) = \int_{W(u,v^-,x)} |w|_x \, dP(w | W(u, v^-, x)),$$

$$= \sum_{k=1}^{\infty} \int_{\{w \in W(u,v^-,x) \wedge |w|_x = k\}} |w|_x \, dP(w | W(u, v^-, x)),$$

$$= \sum_{k=1}^{\infty} k \int_{\{w \in W(u,v^-,x) \wedge |w|_x = k\}} dP(w | W(u, v^-, x)),$$

$$= \sum_{k=1}^{\infty} k \, P(\{w \in W(u, v^-, x) \wedge |w|_x = k\} | W(u, v^-, x)),$$

$$\leq \sum_{k=1}^{\infty} k \, P(\{w : |w|_x = k\} | W(u, v^-, x)),$$

$$= \sum_{k=1}^{\infty} k \, \frac{P(\{w : |w|_x = k \wedge w \in W(u, v^-, x)\})}{P(W(u, v^-, x))},$$

$$\leq \sum_{k=1}^{\infty} k(1 - \alpha)^k \frac{1}{P(W(u, v^-, x))},$$

$$= \frac{1}{\alpha} \left(\frac{1}{\alpha} - 1 \right) \frac{1}{P(W(u, v^-, x))}. \tag{69}$$

We have used Fubini's theorem for the second equality. Then we have:

$$\hat{H}(u, v^-, x) \, P(W(u, v^-, x) | W(u, x)) \leq \frac{1}{\alpha} \left(\frac{1}{\alpha} - 1 \right) \frac{P(W(u, v^-, x) | W(u, x))}{P(W(u, v^-, x))},$$

$$= \frac{1}{\alpha} \left(\frac{1}{\alpha} - 1 \right) \frac{1}{P(W(u, x))}. \tag{70}$$

$P(W(u, x))$ is at least the probability of precisely walking the shortest path from u to x, which has length $d(u, x) \leq 2r$ since u and x are both in $B_r(v)$. This gives us the following:

$$P(W(u, x)) \geq \left(\frac{1 - \alpha}{\Delta} \right)^{d(u,x)},$$

$$\geq \left(\frac{1 - \alpha}{\Delta} \right)^{2r} . \tag{71}$$

Combining this with Equation 70, we have:

$$\hat{H}(u, v^-, x) \, P(W(u, v^-, x) | W(u, x)) \leq \frac{1}{\alpha} \left(\frac{1}{\alpha} - 1 \right) \left(\frac{\Delta}{1 - \alpha} \right)^{2r} . \tag{72}$$

Combining with Equations 62 and 67, we have:

$$H(u, x) \leq \frac{1}{\alpha} + \frac{1}{\alpha} \left(\frac{\Delta}{1 - \alpha} \right)^{d(v,x)} + \frac{1}{\alpha} \left(\frac{1}{\alpha} - 1 \right) \left(\frac{\Delta}{1 - \alpha} \right)^{2r} , \tag{73}$$

for any $\alpha \in (0, 1)$. Notice that, while this bound is for the case $W(u, v^-, x) \neq \emptyset$, it subsumes the bound for the case $W(u, v^-, x) = \emptyset$ (Equation 67). Then, using $d(v, x) \leq r$, we get Equation 54. This completes the proof for random restarts.

We now prove for periodic restarts. If the walk starting at u reaches x before restarting at v, the steps taken is clearly less than k. If the walk starting at u restarts at v at step k before reaching x, it now needs to reach x from v, while restarting at v at every k steps. Let T be the steps taken to reach x from v. Let T' be the number of steps until the walk restarts at v, then exactly follows the shortest path from v to x for the first time, and then restarts at v. It is clear that $\mathbb{E}[T] \leq \mathbb{E}[T']$. Then, we split the walk of length T' into N trials, where each trial consists of restarting at v and walking k steps until the next restart. A trial is successful if it immediately walks the shortest path from v to x. Then N is the number of trials until we get a success, and it follows geometric distribution with probability of success at least $(1/\Delta)^{d(v,x)}$ only for $k \geq d(v, x)$, and zero for $k < d(v, x)$ since the walk cannot reach x before restart. Hence, its expectation is at most $\Delta^{d(v,x)}$ for $k \geq d(v, x)$, and we have $\mathbb{E}[T] \leq \mathbb{E}[T'] = k\mathbb{E}[N] \leq k\Delta^{d(v,x)}$. Adding the k steps until the first restart at v, we have:

$$H(u, x) \leq k + k\Delta^{d(v,x)}, \tag{74}$$

for any $k \geq d(v, x)$. Using $d(v, x) \leq r$, we get Equation 55. This completes the proof. □

We now extend Lemma A.4 to edges. Let $H(u, (x, y))$ be the expected number of steps of a random walk starting at u takes until traversing an edge (x, y) by $x \to y$. We show that $H(u, (x, y))$ is finite for any u and adjacent x, y in $B_r(v)$ for any nonzero restart probability α or restart period $k \geq r + 1$.

Lemma A.5. *For any u and adjacent x, y in $B_r(v)$, $H(u, (x, y))$ is bounded by the following:*

$$H(u, (x, y)) \leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{r+1} + \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r+1}, \tag{75}$$

if the random walk restarts at v with any probability $\alpha \in (0, 1)$, and:

$$H(u, (x, y)) \leq k + k\Delta^{r+1}, \tag{76}$$

if the random walk restarts at v with any period $k \geq r + 1$.

Proof. The proof is almost identical to Lemma A.4, except the target x of reaching is substituted by (x, y) in the direction of $x \to y$, and all arguments that use the shortest path from u or v to x instead use the shortest path to x postfixed by $x \to y$, which adds $+1$ to several terms in the bounds. □

We are now ready to prove Theorem 2.5.

Theorem 2.5. *In Theorem 2.4, if the random walk restarts at v with any nonzero probability α or any period $k \geq r + 1$, the vertex and edge cover times of $B_r(v)$ are always finite.*

Proof. The proof is inspired by the spanning tree argument of Aleliunas et al. (1979). Let us consider a depth first search of $B_r(v)$ starting from v. We denote by T the resulting spanning tree with vertices $V(T) = V(B_r(v))$. We consider the expected time for a random walk starting at v to visit every vertex in the precise order visited by the depth first search by traversing each edge twice. It is clear that this upper-bounds the vertex cover time of $B_r(v)$ starting at v (Equation 18):

$$C_V(B_r(v)) \leq \sum_{(x,y) \in E(T)} [H(x, y) + H(y, x)]. \tag{77}$$

Then, using the bounds from Lemma A.4, the property of spanning trees $|E(T)| = |V(T)| - 1$, and the fact that $|V(T)| = |V(B_r(v))| \leq \Delta^r$ from bounded degree, we obtain:

$$C_V(B_r(v)) \leq 2(\Delta^r - 1)\left(\frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^r + \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r}\right), \tag{78}$$

if the random walk restarts at v with any probability $\alpha \in (0, 1)$, and:

$$C_V(B_r(v)) \leq 2(\Delta^r - 1)(k + k\Delta^r), \tag{79}$$

if the random walk restarts at v with any period $k \geq r$. This completes the proof for the vertex cover time. For the edge cover time, we consider the expected time for a random walk starting at v to visit every edge in the precise order discovered[7] by the depth first search by traversing each edge twice. It is clear that this upper-bounds the edge cover time of $B_r(v)$ starting at v (Equation 19):

$$C_E(B_r(v)) \leq \sum_{(x,y) \in E(B_r(v))} [H(x, (x,y)) + H(y, (y,x))]. \tag{80}$$

Then, using Lemma A.4 and the fact that $|E(B_r(v))| \leq \Delta^{2r} - 1$ from bounded degree, we obtain:

$$C_E(B_r(v)) \leq 2(\Delta^{2r} - 1)\left(\frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{r+1} + \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r+1}\right), \tag{81}$$

if the random walk restarts at v with any probability $\alpha \in (0,1)$, and:

$$C_E(B_r(v)) \leq 2(\Delta^{2r} - 1) \cdot (k + k\Delta^{r+1}), \tag{82}$$

if the random walk restarts at v with any period $k \geq r + 1$. This completes the proof. $\square$

While our proof shows finite bounds for the cover times, it is possible that they can be made tighter, for instance based on Zuckerman (1991). We leave improving the bounds as a future work.

A.9.7 PROOF OF THEOREM 3.2 (SECTION 3.1)

We recall universal approximation of graph-level functions in probability (Definition 3.1):

Definition 3.1. *We say $X_\theta(\cdot)$ is a universal approximator of graph-level functions in probability if, for all invariant functions $\phi : \mathbb{G}_n \to \mathbb{R}$ for a given $n \geq 1$, and $\forall \epsilon, \delta > 0$, there exist choices of length l of the random walk and network parameters θ such that the following holds:*

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] > 1 - \delta, \quad \forall G \in \mathbb{G}_n. \tag{83}$$

We remark that an RWNN $X_\theta(\cdot)$ is composed of a random walk algorithm, a recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$, and a reader neural network $f_\theta : \mathbf{z} \mapsto \hat{\mathbf{y}} \in \mathbb{R}$.

Intuitively, if the record $\mathbf{z}$ of the random walk always provides complete information of the input graph G, we may invoke universal approximation of f_θ to always obtain $|\phi(G) - f_\theta(\mathbf{z})| < \epsilon$, and thus $|\phi(G) - X_\theta(G)| < \epsilon$. However, this is not always true as the random walk may e.g. fail to visit some vertices of G, in which case the record $\mathbf{z}$ would be incomplete. As we show below, this uncertainty leads to the probabilistic bound $> 1 - \delta$ of the approximation.

Let us denote the collection of all possible random walk records as $\{\mathbf{z}\} := \text{Range}(q)$, and consider a decoding function $\psi : \{\mathbf{z}\} \to \mathbb{G}_n$ that takes the record $\mathbf{z} := q(v_0 \to \cdots \to v_l, G)$ of a given random walk $v_{[\cdot]}$ and outputs the graph $\psi(\mathbf{z}) \in \mathbb{G}_n$ composed of all recorded vertices $V(H) := \{\text{id}(v_t) : v_t \in \{v_0, ..., v_l\}\}$ and all recorded edges $E(H) \subset V(H) \times V(H)$. We show the following lemma:

Lemma A.6. *Let $G_\mathbf{z}$ be the subgraph of G whose vertices and edges are recorded by $\mathbf{z}$. Then the graph $\psi(\mathbf{z})$ decoded from the record $\mathbf{z}$ is isomorphic to $G_\mathbf{z}$ through the namespace $\text{id}(\cdot)$:*

$$G_\mathbf{z} \overset{\text{id}}{\simeq} \psi(\mathbf{z}). \tag{84}$$

Furthermore, the decoded graph $\psi(\mathbf{z})$ reconstructs G up to isomorphism, that is,

$$G \overset{\text{id}}{\simeq} \psi(\mathbf{z}), \tag{85}$$

if the recording function $q(\cdot)$ and the random walk $v_{[\cdot]}$ satisfies either of the following:

- *$q(\cdot)$ uses anonymization, and $v_{[\cdot]}$ has traversed all edges of G.*

- *$q(\cdot)$ uses anonymization and named neighbors, and $v_{[\cdot]}$ has visited all vertices of G.*

---

[7]We remark that depth first search discovers all edges of a graph, while not necessarily visiting all of them.

Proof. Equation 84 is straightforward from the fact that the namespace $\mathrm{id}(\cdot)$ defines a bijection from $V(G_{\mathbf{z}})$ to $[|V(G_{\mathbf{z}})|]$, and the recording function uses names $\mathrm{id}(v_t)$ to record vertices and edges. Equation 85 is satisfied when all vertices and edges of G have been recorded, i.e., $G_{\mathbf{z}} = G$, which is possible either when the random walk has traversed all edges of G, or when it has traversed all vertices of G and named neighbors are used to record the induced subgraph $G[V(G)] = G$. $\qquad\square$

We further remark Markov's inequality for any nonnegative random variable T and $a > 0$:

$$\mathrm{Prob}[T \geq a] \leq \frac{\mathbb{E}[T]}{a}. \tag{86}$$

We are now ready to prove Theorem 3.2.

Theorem 3.2. *An RWNN $X_\theta(\cdot)$ with a sufficiently powerful f_θ is a universal approximator of graph-level functions in probability (Definition 3.1) if it satisfies either of the below:*

- *It uses anonymization to record random walks of lengths $l > C_E(G)/\delta$.*

- *It uses anonymization and named neighbors to record walks of lengths $l > C_V(G)/\delta$.*

Proof. Instead of directly approximating the target function $\phi : \mathbb{G}_n \to \mathbb{R}$, it is convenient to define a proxy target function on random walk records $\phi' : \{\mathbf{z}\} \to \mathbb{R}$ where $\{\mathbf{z}\} := \mathrm{Range}(q)$ as follows:

$$\phi' := \phi \circ \psi, \tag{87}$$

where $\psi : \{\mathbf{z}\} \to \mathbb{G}_n$ is the decoding function of walk records. Then, for a given $\mathbf{z}$, we have:

$$G \simeq \psi(\mathbf{z}) \quad\implies\quad \phi(G) = \phi'(\mathbf{z}), \tag{88}$$

which is because ϕ is an invariant function, so $\phi(G) = \phi(\psi(\mathbf{z})) = \phi \circ \psi(\mathbf{z}) = \phi'(\mathbf{z})$. Then we have:

$$\mathrm{Prob}[G \simeq \psi(\mathbf{z})] \leq \mathrm{Prob}[|\phi(G) - \phi'(\mathbf{z})| = 0]. \tag{89}$$

We now invoke universality of f_θ to approximate ϕ'. If f_θ is a universal approximator of functions on its domain $\{\mathbf{z}\} := \mathrm{Range}(q)$, for any $\epsilon > 0$ there exists a choice of θ such that the below holds:

$$|\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon, \quad \forall \mathbf{z} \in \mathrm{Range}(q). \tag{90}$$

Combining Equations 89 and 90, we have:

$$\mathrm{Prob}[G \simeq \psi(\mathbf{z})] \leq \mathrm{Prob}[|\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon]. \tag{91}$$

We remark triangle inequality of distances on $\mathbb{R}$, for a given $\mathbf{z}$:

$$|\phi(G) - f_\theta(\mathbf{z})| \leq |\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})|, \tag{92}$$

which implies, for a given $\mathbf{z}$:

$$|\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon \quad\implies\quad |\phi(G) - f_\theta(\mathbf{z})| < \epsilon, \tag{93}$$

and hence:

$$\mathrm{Prob}[|\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon] \leq \mathrm{Prob}[|\phi(G) - f_\theta(\mathbf{z})| < \epsilon]. \tag{94}$$

Combining Equations 91 and 94, we have:

$$\mathrm{Prob}[|\phi(G) - f_\theta(\mathbf{z})| < \epsilon] \geq \mathrm{Prob}[G \simeq \psi(\mathbf{z})], \tag{95}$$

which can be written as follows:

$$\mathrm{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] \geq \mathrm{Prob}[G \simeq \psi(\mathbf{z})]. \tag{96}$$

We now consider the probability of the event $G \simeq \psi(\mathbf{z})$ based on Lemma A.6. We first consider the case where the recording function $q(\cdot)$ uses anonymization. In this case, $G \simeq \psi(\mathbf{z})$ is achieved if the random walk of length l has traversed all edges of G. Let $T_E(G, v_0)$ be the number of steps that a random walk starting at v_0 takes until traversing all edges of G. Since the edge cover time $C_E(G)$ is its expectation taken at the worst possible starting vertex (Equation 17), we have the following:

$$\mathbb{E}[T_E(G, v_0)] \leq C_E(G), \quad \forall v_0 \in V(G), \tag{97}$$

which leads to the following from Markov's inequality (Equation 86):

$$\text{Prob}[T_E(G, v_0) < l] \geq 1 - \frac{\mathbb{E}[T_E(G)]}{l} \geq 1 - \frac{C_E(G)}{l}. \tag{98}$$

For a given random walk $v_0 \to \cdots \to v_l$ and its record $\mathbf{z}$, the following holds:

$$T_E(G, v_0) < l \quad \implies \quad G \simeq \psi(\mathbf{z}), \tag{99}$$

which implies the following:

$$\text{Prob}[T_E(G, v_0) < l] \leq \text{Prob}[G \simeq \psi(\mathbf{z})]. \tag{100}$$

Combining Equations 96, 98, and 100, we have:

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] \geq 1 - \frac{C_E(G)}{l}. \tag{101}$$

Therefore, for any $\delta > 0$, if we choose $l > C_E(G)/\delta$ we would have the following:

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] > 1 - \delta. \tag{102}$$

This completes the proof for anonymization. The proof is identical for the recording function that uses anonymization and named neighbors, except that the edge cover time is changed to the vertex cover time $C_V(G)$ (Equation 16). This is because named neighbor recording automatically records the induced subgraph of visited vertices, thus visiting all vertices implies recording all edges, $G[V(G)] = G$. □

A.9.8 PROOF OF THEOREM 3.4 (SECTION 3.1)

Theorem 3.4. *An RWNN $X_\theta(\cdot)$ with a sufficiently powerful f_θ and any nonzero restart probability α or restart period $k \geq r + 1$ is a universal approximator of vertex-level functions in probability (Definition 3.3) if it satisfies either of the below for all $B_r(v) \in \mathbb{B}_r$:*

- *It uses anonymization to record random walks of lengths $l > C_E(B_r(v))/\delta$.*

- *It uses anonymization and named neighbors to record walks of lengths $l > C_V(B_r(v))/\delta$.*

Proof. The proof is almost identical to Theorem 3.2, except $G \in \mathbb{G}_n$ are substituted by $B_r(v) \in \mathbb{B}_r$, and the decoding function $\psi : \{\mathbf{z}\} \to \mathbb{B}_r$ is defined to ignore all recorded vertices $\text{id}(x)$ whose shortest path distance from the starting vertex $\text{id}(v) = \text{id}(v_0) = 1$ exceeds r. The latter is necessary to restrict the range of the decoding function ψ to $\mathbb{B}_r$. In addition, any nonzero restart probability α or restart period $k \geq r + 1$ is sufficient to make the cover times $C_E(B_r(v))$ and $C_V(B_r(v))$ finite (Theorem 2.5), thereby guaranteeing the existence of a finite choice of l. □

A.9.9 PROOF OF THEOREM 3.5 (SECTION 3.2)

Theorem 3.5. *The simple RWNN outputs $\mathbf{h}^{(l)} \to \mathbf{x}^\top \boldsymbol{\pi}$ as $l \to \infty$.*

Proof. Since G is connected and non-bipartite, the uniform random walk on it defines an ergodic Markov chain with a unique stationary distribution $\boldsymbol{\pi}$. The limiting frequency of visits on each vertex v is precisely the stationary probability $\boldsymbol{\pi}_v$. Since the model reads $\mathbf{x}_{v_0} \to \cdots \to \mathbf{x}_{v_l}$ by average pooling, the output is given by weighted mean $\sum_v \boldsymbol{\pi}_v \mathbf{x}_v$ which is $\mathbf{x}^\top \boldsymbol{\pi}$. □

A.9.10 PROOF OF THEOREM 3.6 (SECTION 3.2)

Theorem 3.6. *Let $\mathbf{h}_u^{(l)}$ be output of the simple RWNN queried with u. Then:*

$$\mathbb{E}\left[\left|\left|\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{x}_v}\right|\right|\right] = \frac{1}{l+1}\left[\sum_{t=0}^{l} P^t\right]_{uv} \to \boldsymbol{\pi}_v \quad as \quad l \to \infty. \tag{103}$$

Proof. Since the model reads $\mathbf{x}_{v_0} \to \cdots \to \mathbf{x}_{v_l}$ by average pooling, the feature Jacobian $|\partial \mathbf{h}_u^{(l)}/\partial \mathbf{x}_v|$ is given as number of visits to the vertex v in the random walk $v_0 \to \cdots \to v_l$ starting at $v_0 = u$, divided by length $l + 1$. Let us denote the expected number of these visits by $J(u, v, l)$. Let $\mathbb{1}_{v_t=v}$ be the indicator function that equals 1 if $v_t = v$ and 0 otherwise. Then we can write $J(u, v, l)$ as:

$$
J(u, v, l) = \mathbb{E}\left[\sum_{t=0}^{l} \mathbb{1}_{v_t=v} | v_0 = u\right],
$$
$$
= \sum_{t=0}^{l} \mathbb{E}[\mathbb{1}_{v_t=v} | v_0 = u]. \tag{104}
$$

We have used linearity of expectations for the second equality. $\mathbb{E}[\mathbb{1}_{v_t=v} | v_0 = u]$ is the probability of being at v at step t given that the walk started at u. This probability is precisely $[P^t]_{uv}$. Therefore:

$$
J(u, v, l) = \sum_{t=0}^{l} \left[P^t\right]_{uv} = \left[\sum_{t=0}^{l} P^t\right]_{uv}, \tag{105}
$$

which gives the equality in Equation 103. Furthermore, since G is connected and non-bipartite, the uniform random walk on it defines an ergodic Markov chain with a unique stationary distribution $\boldsymbol{\pi}$. The limiting frequency of visits on vertex v is precisely the stationary probability $\boldsymbol{\pi}_v$, which gives the convergence in Equation 103. This completes the proof. $\qquad\square$

A.10   EXTENDED RELATED WORK (SECTION 4)

Anonymized random walks (Micali & Zhu, 2016)   The initial work on anonymization by Micali & Zhu (2016) has stated an important result, that a sufficiently long anonymized walk starting from a vertex v encodes sufficient information to reconstruct the local subgraph $B_r(v)$ up to isomorphism. While the focus of Micali & Zhu (2016) was a probabilistic graph reconstruction algorithm that uses a set of independent anonymized walks and accesses the oracle set of all possible anonymized walks, we adopt the idea in a neural processing context to acquire universality in probability (Section 3.1). In addition, the invariance property of anonymization has rarely been noticed formally in the literature, which is our key motivation for using it, on top of being able to recover the whole graph (Section 2).

In-depth comparison with CRaWl (Tönshoff et al., 2023)   Our approach is related to CRaWl in two key aspects: **(1)** identity and connectivity encodings of CRaWl contains analogous information to anonymization and named neighbor recording, respectively, and **(2)** CRaWl uses 1D CNNs as the reader NN. A key technical difference lies in the first part. The identity and connectivity encodings of CRaWl are defined within a fixed window size (denoted s in (Tönshoff et al., 2023)), which puts a locality constraint on the recorded information. Precisely, the encodings at step t can encode the information of the walk from step $t - s$ to t (precisely, its induced subgraph), referred to as a walklet. The window size s is a hyperparameter that controls the expressive power of CRaWl, and this dependency makes CRaWl non-universal (Section 3.2, (Tönshoff et al., 2023)). Our choice of anonymization and named neighbor recording are not under such a local constraint, and they encode the full information of a given walk globally. This property underlies our universality results in Section 3.1, which also naturally motivates our choice of universal reader NNs, e.g. a transformer language model, which were not explicitly considered in CRaWl.

Random walks on graphs   Our work builds upon theory of random walks on graphs, i.e., Markov chains on discrete spaces. Their statistical properties such as hitting and mixing times have been well-studied (Aleliunas et al., 1979; Lovász, 1993; Coppersmith et al., 1996; Feige, 1995; Oliveira, 2012; Peres & Sousi, 2015), and our Section 2 is related to vertex cover time (Aleliunas et al., 1979; Kahn et al., 1989; Ding et al., 2011; Abdullah, 2012), edge cover time (Zuckerman, 1991; Bussian, 1996; Panotopoulou, 2013; Georgakopoulos & Winkler, 2014), and improving them, using local degree information (Ikeda et al., 2009; Abdullah et al., 2015; David & Feige, 2018), non-backtracking (Alon et al., 2007; Kempton, 2016; Arrigo et al., 2019; Fasino et al., 2021), or restarts (Dumitriu et al., 2003; McNew, 2013; Janson & Peres, 2012). Our work is also inspired by graph algorithms based on random walks, such as anonymous observation (Micali & Zhu, 2016), sublinear algorithms (Dasgupta et al., 2014; Chiericetti et al., 2016; Ben-Hamou et al., 2018; Bera & Seshadhri, 2020), and personalized PageRank for search (Page et al., 1999) and message passing (Gasteiger et al., 2019). While we adopt their techniques to make our walks and their records well-behaved, the difference is that we use a non-linear deep neural network to process the records and directly make predictions. Our method is also related to label propagation algorithms (Zhu & Ghahramani, 2002; Zhu, 2005; Grady, 2006) that perform transductive learning on graphs based on random walks, as we have discussed in Section 5.3.

Over-smoothing and over-squashing   Prior work on over-smoothing and over-squashing of MPNNs (Barceló et al., 2020; Topping et al., 2022; Giovanni et al., 2023; Giraldo et al., 2023; Black et al., 2023; Nguyen et al., 2023; Park et al., 2023; Wu et al., 2023) often make use of structural properties of graphs, such as effective resistance (Doyle & Snell, 1984; Chandra et al., 1997), Laplacian eigen-spectrum (Lovász, 1993; Spielman, 2019), and discrete Ricci curvatures (Ollivier, 2009; Devriendt & Lambiotte, 2022). Interestingly, random walks and their statistical properties are often closely related to these properties, indicating some form of parallelism between MPNNs and RWNNs. This has motivated our analysis in Section 3.2, where we transfer the prior results on over-smoothing and over-squashing of MPNNs based on these properties into the results on our approach.

Unlike the MPNNs considered in the above prior work, AMP (Errica et al., 2023) mitigates over-smoothing and over-squashing by learning to prune messages at each step (among its other techniques). This brings it close to attention-based MPNNs, which can be understood as modifying the computation (P in Section 3.2) over topology. Indeed, our description of over-smoothing and over-squashing in MPNNs in Section 3.2 mainly applies to message passing using fixed weights, and attention falls out of this framework, which is a shared limitation with prior work (Topping et al., 2022; Giraldo et al., 2023; Black et al., 2023; Nguyen et al., 2023). Formal propagation analysis for attention-based

MPNNs is currently in its infancy (Wu et al., 2023), but we believe it is interesting since, based on our parallelism between MPNNs and RWNNs, it may tell us about how to adapt the walk probabilities adaptively based on input data. We leave investigating this as future work.

Language models for learning on graphs   While not based on random walks, there have been prior attempts on applying language models for problems on graphs (Wang et al., 2023; Chen et al., 2023; Zhao et al., 2023; Fatemi et al., 2023; Ye et al., 2024), often focusing on prompting methods on problems involving simulations of graph algorithms. We take a more principle-oriented approach based on invariance and expressive power, and thus demonstrate our approach mainly on the related tasks, e.g. graph separation. We believe extending our work to simulating graph algorithms requires a careful treatment (Weiss et al., 2021; Delétang et al., 2023; de Luca & Fountoulakis, 2024; Sanford et al., 2024) and plan to investigate it as future work.

A.11   LIMITATIONS AND FUTURE WORK

Our current main limitation is the cost of performing training and inference if using language models, especially on long walks. While random walk sampling and text recording can be done efficiently, the main computational bottleneck comes from the use of language models as the reader NN. For example, in Section 5.3, each test prediction of RWNN-Llama-3-70b takes around 0.410 ± 0.058 seconds on a $4\times$ H100 machine (RWNN-Llama-3-8b is faster, 0.116 ± 0.064 seconds per prediction). The inference time of standard MPNNs are much shorter, e.g., a 3-layer GAT with 16 hidden dimensions takes around one second for all test vertices of ogbn-arxiv on a V100 GPU (Wang et al., 2019). Overcoming this issue is an important future direction. We remark that, if language model embeddings are used and fine-tuned, the training cost of MPNNs can also be substantial, e.g., recent work has reported that prior methods for training an MPNN jointly with language model embeddings on arXiv take 25-46 hours on a $6\times$ RTX 3090 machine (Table 3 of Zhu et al. (2024)).

On the theory side, we focus on cover times of random walks in our analysis (Section 3.1 and Appendix A.8), which corresponds to the worst-case length bounds required for universal approximation. Yet, in practice, there is a substantial amount of useful information that can be already recovered from possibly non-covering walks. Examples include spectral properties (Benjamini et al., 2006), number of edges and mixing time (Ben-Hamou et al., 2018), and triangle counts (Bera & Seshadhri, 2020), among others studied in the field of sublinear algorithms. These results suggest that shorter walks may suffice in practice for RWNNs, if the task at hand requires properties that can be estimated without seeing the whole graph, and the reader NN can learn to recover them (e.g., leveraging universality). Understanding this aspect better is an important direction for future work.

Another question for future work is whether we can train a language model on a large pseudocorpus of random walks (Klubicka et al., 2019; 2020) to build a foundation model for graphs which is language-compatible. We plan to investigate this direction in the future.