Revisiting Random Walks for Learning on Graphs

Jinwoo Kim [1] **Olga Zaghen** [2 3 *] **Ayhan Suleymanzade** [1 *] **Youngmin Ryou** [1] **Seunghoon Hong** [1]

Editors: *S. Vadgama, E.J. Bekkers, A. Pouplin, S.O. Kaba, H. Lawrence, R. Walters, T. Emerson, H. Kvinge, J.M. Tomczak, S. Jegelka*

Abstract

We revisit a simple idea for machine learning on graphs, where a random walk on a graph produces a machine-readable record, and this record is processed by a deep neural network to directly make vertex-level or graph-level predictions. We call these stochastic machines **random walk neural networks**, and show that we can design them to be isomorphism invariant while capable of universal approximation of graph functions in probability. Notably, almost any record of random walk guarantees probabilistic invariance as long as vertices are anonymized. This enables us to record random walks in plain text and adopt a language model to read these text records to solve graph tasks. We demonstrate random walk neural networks based on pre-trained language models on several hard problems on graphs: separating strongly regular graphs where 3-WL fails, counting substructures, and transductive classification on arXiv citation network without training. Code is at this link.

1. Introduction

The most widely used class of neural networks on graphs are message passing neural networks where each vertex keeps a feature vector and updates it by propagating and aggregating messages. These networks benefit in efficiency by respecting the natural symmetries of graph functions, namely invariance to graph isomorphism (Chen et al., 2019). On the other hand, message passing can be viewed as implementing iterations of the 1-WL test, and thus their expressive power is not stronger (Xu et al., 2019). Also, their inner working is tied to the topology of the input graph, which is related to over-smoothing, over-squashing, and under-reaching of features (Topping et al., 2022; Giraldo et al., 2023).

In this work, we (re)visit a simple alternative idea for learning on graphs, where a random walk on a graph produces a machine-readable record, and this record is processed by a deep neural network that directly makes vertex-level or graph-level predictions. We call these stochastic machines **random walk neural networks**. Using random walks for graph learning has received practical interest due to their scalability and compatibility with sequence learning methods. This has led to empirically successful algorithms such as DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) which use skip-gram on random walks, and CRaWl (Tönshoff et al., 2023) which uses convolutions on random walks. Yet, principled understanding and design of random walk neural networks have been studied less.

We show that random walk neural networks can be designed from principle to be isomorphism invariant while capable of universal approximation of graph functions in probability. Specifically, **(1)** almost any record of random walks guarantees invariance as long as the vertices are anonymized, and there is no constraint on the neural network that processes the records, **(2)** universality is achieved if the neural network on records is powerful enough, and the random walk covers the whole graph of interest with a high probability, and **(3)** over-smoothing is avoided by construction while over-squashing manifests as probabilistic under-reaching. The findings motivate us to record random walks in plain text and adopt a pre-trained language model to read these records to solve graph tasks. They also give us principled guidance on how to design the random walk algorithm, such as using restarts for vertex-level tasks on a large graph.

Empirically, our model based on DeBERTa (He et al., 2021) learns separation of strongly regular graphs with perfect accuracy whereas 3-WL test fails, and it also successfully learns to count 8-cycles at graph- and vertex-level. We also show that our approach can turn transductive classification into an in-context learning problem by recording labeled vertices during walks. To demonstrate this, we apply frozen Llama 3 (meta llama, 2024) directly to arXiv network with 170k vertices, and show that it outperforms message passing as well as zero- and few-shot baselines even without training. This verifies that our approach allows a pure language model to understand and leverage graph structures.

[*]Equal contribution [1]KAIST [2]University of Amsterdam [3]Work done during an internship at School of Computing, KAIST. Correspondence to: Seunghoon Hong <seunghoon.hong@kaist.ac.kr>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at the 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024.

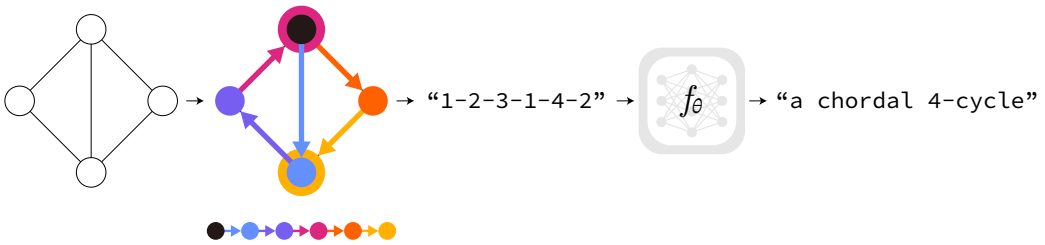


Figure 1: A random walk neural network that reads text record using a language model.

2. Random Walk Neural Networks

We define a random walk neural network as a randomized function $X_\theta(\cdot)$ that takes a graph G as input and outputs a random variable $X_\theta(G)$ on the output space $\mathbb{R}^d$.[4] It can be optionally queried with an input vertex v which gives a random variable $X_\theta(G, v)$. It consists of the following:

1. **Random walk algorithm** that produces l steps of vertex transitions $v_0 \to \cdots \to v_l$ on the graph G. If input vertex v is given, we fix the starting vertex by $v_0 = v$.
2. **Recording function** $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ that produces a machine-readable record $\mathbf{z}$ of the random walk. It may access the graph G to record auxiliary information such as attributes.
3. **Reader neural network** $f_\theta : \mathbf{z} \mapsto \hat{\mathbf{y}}$ that processes record $\mathbf{z}$ and outputs a prediction $\hat{\mathbf{y}}$ in $\mathbb{R}^d$. It is the only trainable component and is not restricted to specific architectures.

We discuss choices of the above for graph-level tasks on finite graphs, and then vertex-level tasks on possibly infinite graphs. Pseudocode and proofs are in Appendix A.4, A.6.

2.1. Graph-Level Tasks

Let $\mathbb{G}$ be the class of undirected, connected, simple graphs[5]. Let $n \geq 1$ and $\mathbb{G}_n$ be the collection of graphs in $\mathbb{G}$ with $\leq n$ vertices. We would like to model a graph-level function $\phi : \mathbb{G}_n \to \mathbb{R}^d$ using a random walk neural network $X_\theta(\cdot)$. It is reasonable to assume that ϕ is isomorphism invariant:

$$\phi(G) = \phi(H), \quad \forall G \simeq H. \tag{1}$$

Incorporating invariance to our model class would offer generalization benefit (Lyle et al., 2020). Since $X_\theta(\cdot)$ is a randomized function, we accept the probabilistic notion of invariance (Bloem-Reddy & Teh, 2020; Cotta et al., 2023):

$$X_\theta(G) \overset{d}{=} X_\theta(H), \quad \forall G \simeq H. \tag{2}$$

[4]While any output type such as text is possible (Figure 1), we explain with vector output for simplicity.

[5]We assume this for simplicity but extending to directed or attributed graphs is possible.

We now claim that we can achieve invariance by properly choosing the random walk algorithm and recording function while not imposing any constraint on reader neural network.

Theorem 2.1. *$X_\theta(\cdot)$ is invariant if its random walk algorithm and recording function are invariant.*

We now describe the components that satisfy Theorem 2.1.

Random walk algorithm A random walk algorithm is invariant if it satisfies the following:

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \quad \forall G \overset{\pi}{\simeq} H, \tag{3}$$

where $v_{[\cdot]}$ is a random walk on G, $u_{[\cdot]}$ is a random walk on H, and $\pi : V(G) \to V(H)$ is the isomorphism from G to H. It turns out that many random walk algorithms in literature are already invariant. Let us write the probability of walking from a vertex u to its neighbor $x \in N(u)$:

$$\mathrm{Prob}[v_t = x | v_{t-1} = u] := \frac{c_G(u, x)}{\sum_{y \in N(u)} c_G(u, y)}, \tag{4}$$

where the function $c_G : E(G) \to \mathbb{R}_+$ assigns positive weights called conductance to edges. If we use constant conductance $c_G(\cdot) = 1$, we have unbiased random walk used in DeepWalk (Perozzi et al., 2014). We can show:

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

$$c_G(u, v) = c_H(\pi(u), \pi(v)), \quad \forall G \overset{\pi}{\simeq} H. \tag{5}$$

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

We favor using degree information as it is cheap while potentially improving the behaviors of walks. In particular, the following conductance called the minimum degree local rule (Abdullah et al., 2015; David & Feige, 2018) is useful:

$$c_G(u, v) := \frac{1}{\min[\deg(u), \deg(v)]}. \tag{6}$$

This conductance is special as it achieves $O(n^2)$ cover time, i.e. the expected time of visiting all n vertices of a graph,

which is optimal among random walks in Equation (4) that use degrees of endpoints (David & Feige, 2018).

In practice, we find adding non-backtracking property that enforces $v_{t+1} \neq v_{t-1}$ is helpful while not adding much cost. Formally, we can extend Theorem 2.2 to second-order walks that include non-backtracking and node2vec walks (Grover & Leskovec, 2016). We leave this in Appendix A.3.

Recording function A recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ takes a random walk and produces a machine-readable record $\mathbf{z}$. We let it have access to the graph G to allow recording auxiliary information such as vertex or edge attributes. A recording function is invariant if it satisfies the below for any given random walk $v_{[\cdot]}$ on G:

$$q(v_0 \to \cdots \to v_l, G) = q(\pi(v_0) \to \cdots \to \pi(v_l), H), \quad \forall G \overset{\pi}{\simeq} H. \quad (7)$$

Invariance requires that $q(\cdot, G)$ produces the same record $\mathbf{z}$ regardless of re-indexing of vertices of G into H. For this, we have to be careful in how we represent each vertex in a walk $v_{[\cdot]}$ as a machine-readable value, and which auxiliary information we record from G. We make two choices:

- **Anonymization.** We name each vertex in a walk with a unique integer, starting from $v_0 \mapsto 1$ and incrementing it based on their order of discovery. For instance, a walk $a \to b \to c \to a$ translates to $1 \to 2 \to 3 \to 1$.
- **Anonymization + named neighborhoods.** While applying anonymization for each vertex v in a walk, we record its neighbors $u \in N(v)$ if they are already named but the edge $v \to u$ has not been recorded yet. For instance, a walk $a \to b \to c \to d$ on a fully-connected graph translates to $1 \to 2 \to 3\langle 1 \rangle \to 4\langle 1, 2 \rangle$, where $\langle \cdot \rangle$ represents the named neighborhoods.

While anonymization was developed in Micali & Zhu (2016) from privacy motivations, we find it useful for invariance. Recording named neighborhoods is useful because whenever a walk visits a set of vertices $S \subseteq V(G)$ it automatically records the entire induced subgraph $G[S]$. As a result, to record all edges of a graph, a walk only has to visit all vertices. It is then possible to choose a random walk algorithm that takes only $O(n^2)$ time to visit all n vertices (see Equation (6)), while traversing all edges, i.e. edge cover time, is $O(n^3)$ in general (Zuckerman, 1991). We now show:

Theorem 2.3. *A recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ that uses either of the two choices is invariant.*

Reader neural network A reader neural network $f_\theta : \mathbf{z} \mapsto \hat{\mathbf{y}}$ processes the record $\mathbf{z}$ of random walk and outputs a prediction $\hat{\mathbf{y}}$. As in Theorem 2.1, there is no invariance constraint imposed on f_θ, and any neural network that accepts the recorded walks and has a sufficient expressive power can be used. This is in contrast to message passing where invariance is hard-coded in feature mixing operations. Also, our record $\mathbf{z}$ can take any format, such as a matrix or a byte stream, as long as f_θ accepts it. It is appealing in this regard to choose the record to be plain text, such as `"1-2-3-1"`, and choose f_θ to be a pre-trained transformer language model. This offers expressive power (Yun et al., 2020) as well as transferable representation (Lu et al., 2022). In experiments, we test both "encoder-only" DeBERTa (He et al., 2021) and "decoder-only" Llama (Touvron et al., 2023).

2.2. Vertex-Level Tasks

We now consider vertex-level tasks. In case of finite graphs, we may simply frame a vertex-level task $(G, v) \mapsto \mathbf{y}$ as a graph-level task $G' \mapsto \mathbf{y}$ where G' is G with v marked.

A more interesting case is when G is an infinite graph that is only locally finite. This simulates learning problems on a large graph e.g. transductive classification, and forces the model to operate independently of the size of the whole G (Micali & Zhu, 2016). In this case, it is reasonable to assume that our target function depends on finite local structures.

Let $r \geq 1$, and $\mathbb{B}_r := \{B_r(v)\}$ be the collection of local balls in G of radius r centered at $v \in V(G)$. We would like to model a vertex-level function $\phi : \mathbb{B}_r \to \mathbb{R}^d$ on G using a random walk neural network $X_\theta(\cdot)$ by querying the vertex of interest, $X_\theta(G, v)$. We assume that ϕ is invariant:

$$\phi(B_r(v)) = \phi(B_r(u)), \quad \forall B_r(v) \simeq B_r(u). \quad (8)$$

The invariance of $X_\theta(\cdot)$ in probability is defined as:

$$X_\theta(G, v) \overset{d}{=} X_\theta(G, u), \quad \forall B_r(v) \simeq B_r(u). \quad (9)$$

We may achieve invariance by choosing the random walk algorithm and recording function similar to the graph-level case[6]. As a modification, we query $X_\theta(G, v)$ with the starting vertex $v_0 = v$ of the random walk so that anonymization informs v to reader neural network by always naming it 1.

Then, we make an important choice of localizing random walks with restarts. That is, we reset a walk to the starting vertex v_0 either with a probability $\alpha \in (0, 1)$ at each step or periodically every k steps. A restarting walk tends to stay around its starting vertex v_0, and was used to implement locality bias in personalized PageRank algorithm for search (Page et al., 1999). This effect is crucial in our context since a walk may drift away from $B_r(v_0)$ before recording all necessary information, which will then take an infinite time to return since G is infinite (McNew, 2013). Restarts make the return to v_0 mandatory, ensuring that $B_r(v_0)$ can be recorded in a finite expected time. We show:

[6] This is assuming that the random walks are localized in $B_r(v)$ and $B_r(u)$, e.g. with restarts.

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

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

3. Analysis

In Section 2, we have described the design of random walk neural networks, primarily relying on the principle of invariance. We now provide in-depth analysis on their expressive power and relations to issues in message passing such as over-smoothing, over-squashing, and under-reaching.

3.1. Expressive Power

We first consider using a random walk neural network $X_\theta(\cdot)$ to universally approximate graph-level functions $\phi(\cdot)$ in probability, as defined in Abboud et al. (2021).

Definition 3.1. $X_\theta(\cdot)$ 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:

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] > 1 - \delta, \quad \forall G \in \mathbb{G}_n. \quad (10)$$

We show that if the walk is long enough and the reader f_θ is powerful e.g. an MLP (Hornik et al., 1989) or a transformer (Yun et al., 2020), $X_\theta(\cdot)$ is capable of graph-level universal approximation. In particular, the length l controls the confidence $1-\delta$ of the approximation, with cover time $C_V(G)$ or edge cover time $C_E(G)$ playing a central role.

Theorem 3.2. *$X_\theta(\cdot)$ with a sufficiently powerful f_θ is a universal approximator of graph-level functions if:*

- *It uses anonymization to record random walks of lengths $l > C_E(G)/\delta$,*
- *or it uses anonymization and named neighborhoods to record walks of lengths $l > C_V(G)/\delta$.*

We remark that, while cover times are $O(n^3)$ for uniform random walks (Aleliunas et al., 1979; Zuckerman, 1991), we use Equation (6) to achieve a cover time of $O(n^2)$, in conjunction with named neighborhoods. Our design hence reduces the length l required for the desired reliability $> 1-\delta$, which improves learning and inference.

Results for vertex-level functions are similar, relating to local cover time $C_V(B_r(v))$ and edge cover time $C_E(B_r(v))$:

Definition 3.3. $X_\theta(\cdot)$ is a universal approximator of vertex-level functions in probability if, for all invariant functions $\phi : \mathbb{B}_r \to \mathbb{R}$ for a given $r \geq 1$, and $\forall \epsilon, \delta > 0$, there exist choices of length l and restart probability α or period k of the random walk and network parameters θ such that:

$$\text{Prob}[|\phi(B_r(v)) - X_\theta(G, v)| < \epsilon] > 1 - \delta, \quad \forall B_r(v) \in \mathbb{B}_r. \quad (11)$$

Theorem 3.4. *$X_\theta(\cdot)$ 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 if:*

- *It uses anonymization to record random walks of lengths $l > C_E(B_r(v))/\delta$,*
- *or it uses anonymization and named neighborhoods to record walks of lengths $l > C_V(B_r(v))/\delta$.*

We remark that restarts are necessary to finitely bound the local cover times of a random walk on an infinite graph. From this fact and Theorem 3.4 we can see that non-restarting walks of finite length cannot provide vertex-level universal approximation in probability, and our design is obligatory.

3.2. Over-smoothing, Over-squashing, Under-reaching

In message passing neural networks, the model operates by passing features over edges and mixing them, and thus their operation is tied to the topology of the input graph. It is now understood how this relates to the issues of over-smoothing, over-squashing, and under-reaching (Barceló et al., 2020; Topping et al., 2022; Giovanni et al., 2023; Giraldo et al., 2023; Nguyen et al., 2023; Wu et al., 2023). We connect these theories and random walk neural networks to verify if similar issues may take place.

Let G be a connected non-bipartite graph, and denote its row-normalized adjacency matrix by P. We consider a simplified message passing model, where the vertex features $\mathbf{h}^{(0)}$ are initialized as some probability vector $\mathbf{x}$, and updated by $\mathbf{h}^{(t+1)} = \mathbf{h}^{(t)}P$. This simple model is often used as a proxy to study the aforementioned issues (Giraldo et al., 2023; Zhao & Akoglu, 2019). For example, over-smoothing happens as the features exponentially converge to a stationary vector $\mathbf{h}^{(l)} \to \boldsymbol{\pi}$ as $l \to \infty$, smoothing out the input $\mathbf{x}$ (Giraldo et al., 2023). Over-squashing and under-reaching occur when a feature $\mathbf{h}_u^{(l)}$ becomes insensitive to distant input $\mathbf{x}_v$. While under-reaching refers to insufficient depth $l < \text{diam}(G)$ (Barceló et al., 2020; Topping et al., 2022), over-squashing refers to features getting overly compressed at bottlenecks of G, even with sufficient depth l. The latter is described by the Jacobian $|\partial \mathbf{h}_u^{(l)} / \partial \mathbf{x}_v| \leq [\sum_{t=0}^{l} P^t]_{uv}$,[7] as the bound often decays exponentially with l (Topping et al., 2022; Black et al., 2023).

[7] This bound is obtained by applying Lemma 3.2 of Black et al. (2023) to our simplified message passing.

What do these results tell us about random walk neural networks? We can see that, while P drives feature mixing in the message passing schema, it can be also interpreted as the transition probability matrix of uniform random walk where P_{uv} is the probability of walking from u to v. This parallelism motivates us to design an analogous, simplified random walk neural network and study its behavior.

We consider a simple random walk neural network that runs a uniform random walk $v_0 \to \cdots \to v_l$, reads the record $\mathbf{x}_{v_0} \to \cdots \to \mathbf{x}_{v_l}$ by averaging, and outputs it as $\mathbf{h}^{(l)}$. Like simple message passing, the model involves l steps of time evolution through P. Yet, while message passing uses P to process features, this model uses P only to obtain a record of input, with feature processing decoupled. Then, over-smoothing does not occur as in simple message passing[8]:

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

Even if the time evolution through P happens fast (i.e. P is rapidly mixing) or with many steps l, the model is resistant to over-smoothing as the input $\mathbf{x}$ always affects the output. In fact, if P is rapidly mixing, we may expect improved behaviors based on Section 3.1 as cover times could reduce.

On the other hand, we show that over-squashing manifests as probabilistic under-reaching:

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

$$\mathbb{E}\left[\left|\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{x}_v}\right|\right] = \frac{1}{l+1}\left[\sum_{t=0}^{l} P^t\right]_{uv} \to \boldsymbol{\pi}_v \quad as \quad l \to \infty. \tag{12}$$

The equation shows that the feature Jacobians are bounded by the sum of powers of P, same as in simple message passing. Both models are subject to over-squashing phenomenon that is similarly formalized, but manifests through different mechanisms. While in message passing the term is related to over-compression of features at bottlenecks (Topping et al., 2022), in random walk neural networks it is related to exponentially decaying probability of reaching a distant vertex v, i.e., probabilistic under-reaching.

In message passing, it is understood that the topology of the input graph inevitably induces a trade-off between over-smoothing and over-squashing (Nguyen et al., 2023; Giraldo et al., 2023). Our results suggest that random walk neural networks avoid the trade-off, and we can focus on overcoming under-reaching e.g. by accelerating the walk, while not worrying much about over-smoothing. Our design such as degree-biasing (Equation (6)) and non-backtracking (Alon et al., 2007) can be understood as achieving this.

[8]While we show not forgetting $\mathbf{x}$ for brevity, we may extend to initial vertex v_0 using its anonymization as 1.

Table 1: SR25 graph separation. ‡ and $^\diamond$ are from Zhao et al. (2022b) and Alvarez-Gonzalez et al. (2024).

Method	Accuracy
GIN (Xu et al., 2019)	6.7%‡
PPGN (Maron et al., 2019)	6.7%‡
GIN-AK+ (Zhao et al., 2022b)	6.7%‡
PPGN-AK+ (Zhao et al., 2022b)	100.0%‡
GIN+ELENE (Alvarez-Gonzalez et al., 2024)	6.7%$^\diamond$
GIN+ELENE-L (ED) (Alvarez-Gonzalez et al., 2024)	100.0%$^\diamond$
AgentNet (Martinkus et al., 2023)	6.7%
CRaWl (Tönshoff et al., 2023)	46.6%
DeBERTa-walk (Ours)	100.0%

Table 2: ogbn-arxiv transductive classification. † denotes using validation label for prediction as in Huang et al. (2020).

Method	Accuracy
node2vec (Hu et al., 2020)	70.07%
GCN (Hu et al., 2020)	71.74%
RevGAT (Li et al., 2021)	74.26%
C&S (Huang et al., 2020)	72.62%
C&S† (Huang et al., 2020)	74.02%
GPT-3.5 zero-shot (He et al., 2023)	73.50%
DeBERTa, fine-tuned (Zhao et al., 2022a)	74.13%
Llama3-8b zero-shot	50.20%
Llama3-8b one-shot	52.49%
Llama3-8b one-shot†	52.54%
Llama3-8b-walk (Ours)	71.10%
Llama3-8b-walk (Ours)†	73.11%
Llama3-70b zero-shot	65.33%
Llama3-70b one-shot†	67.57%
Llama3-70b-walk (Ours)†	**74.75%**

4. Experiments

Due to space limits, we present here a subset of the results. Full experimental results are given in Appendix A.2, A.5.

SR25 We verify claims on expressive power using SR25 graph separation dataset (Balcilar et al., 2021) where the task is classifying training graphs and 3-WL test fails. We use degree-biasing in Equation (6) with non-backtracking, recording function with anonymization and named neighborhoods, and reader neural network fine-tuned from DeBERTa-base language model. Table 1 shows that our model achieves perfect accuracy for the first time in walk-based methods.

arXiv ogbn-arxiv (Hu et al., 2020) is a citation network of 169,343 papers with text attributes, and the task is 40-way transductive vertex classification using labeled vertices. We use restarting random walks with $\alpha = 0.7$ and modify the recording function to include labels in case of labeled vertices. The resulting record includes a number of input-label pairs, such that we can frame transductive classification as in-context learning problem and directly use Llama 3 (meta llama, 2024) language models. In Table 2, our model outperforms message passing networks, vertex-wise language models, and one-shot baselines using 40 labeled examples, showing that our method allows a pure language model to understand and leverage graph structures without training.

Conclusion We presented random walk neural networks, stochastic machines that process a graph as a random walk. With proper design choices, they achieve invariance and universality in probability with unconstrained neural networks. We discuss limitations and future work in Appendix A.7.

Acknowledgements

This work was supported in part by the National Research Foundation of Korea (RS-2024-00351212) and the IITP grant (2022-0-00926 and RS-2022-II220959) funded by the Korean government (MSIT). We thank the anonymous reviewers for their insightful comments.

References

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021. (pages 4, 11, 12, 13)

Abdullah, M. The cover time of random walks on graphs. *arXiv*, 2012. (page 11)

Abdullah, M. A., Cooper, C., and Draief, M. Speeding up cover time of sparse graphs using local knowledge. In *IWOCA*, 2015. (pages 2, 11)

Aleliunas, R., Karp, R. M., Lipton, R. J., Lovász, L., and Rackoff, C. Random walks, universal traversal sequences, and the complexity of maze problems. In *Annual Symposium on Foundations of Computer Science*, 1979. (pages 4, 11, 37)

Alon, N., Benjamini, I., Lubetzky, E., and Sodin, S. Non-backtracking random walks mix faster. *Communications in Contemporary Mathematics*, 2007. (pages 5, 11, 15)

Alvarez-Gonzalez, N., Kaltenbrunner, A., and Gómez, V. Improving subgraph-GNNs via edge-level ego-network encodings. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. (pages 5, 12)

Arrigo, F., Higham, D. J., and Noferini, V. Non-backtracking pagerank. *J. Sci. Comput.*, 2019. (page 11)

Balcilar, M., Héroux, P., Gaüzère, B., Vasseur, P., Adam, S., and Honeine, P. Breaking the limits of message passing graph neural networks. In *ICML*, 2021. (pages 5, 12)

Barceló, P., Kostylev, E. V., Monet, M., Pérez, J., Reutter, J. L., and Silva, J. P. The logical expressiveness of graph neural networks. In *ICLR*, 2020. (pages 4, 11)

Ben-Hamou, A., Oliveira, R. I., and Peres, Y. Estimating graph parameters via random walks with restarts. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018. (pages 11, 41)

Benjamini, I., Kozma, G., Lovasz, L., Romik, D., and Tardos, G. Waiting for a bat to fly by (in polynomial time). *Combinatorics, Probability and Computing*, 2006. (page 41)

Bera, S. K. and Seshadhri, C. How to count triangles, without seeing the whole graph. In *KDD*, 2020. (page 11)

Black, M., Wan, Z., Nayyeri, A., and Wang, Y. Understanding oversquashing in gnns through the lens of effective resistance. In *ICML*, 2023. (pages 4, 11)

Bloem-Reddy, B. and Teh, Y. W. Probabilistic symmetries and invariant neural networks. *J. Mach. Learn. Res.*, 2020. (pages 2, 11)

Bronstein, M. M., Bruna, J., Cohen, T., and Velickovic, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv*, 2021. (page 11)

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020. (page 14)

Bussian, E. R. *Bounding the edge cover time of random walks on graphs*. Georgia Institute of Technology, 1996. (page 11)

carnage. Expected hitting time for simple random walk from origin to point (x,y) in 2d-integer-grid. MathOverflow, 2012. URL https://mathoverflow.net/questions/112470. [Online:] https://mathoverflow.net/questions/112470. (page 32)

Chandra, A. K., Raghavan, P., Ruzzo, W. L., Smolensky, R., and Tiwari, P. The electrical resistance of a graph captures its commute and cover times. *Comput. Complex.*, 1997. (page 11)

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv*, 2023a. (page 41)

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. In *NeurIPS*, 2019. (pages 1, 12)

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? In *NeurIPS*, 2020. (page 13)

Chen, Z., Mao, H., Li, H., Jin, W., Wen, H., Wei, X., Wang, S., Yin, D., Fan, W., Liu, H., and Tang, J. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor.*, 2023b. (page 11)

Chenebaux, M. graph-walker: Fastest random walks generator on networkx graphs. https://github.com/kerighan/graph-walker, 2020. (page 12)

Chiericetti, F., Dasgupta, A., Kumar, R., Lattanzi, S., and Sarlós, T. On sampling nodes in a network. In *WWW*, 2016. (page 11)

Coppersmith, D., Feige, U., and Shearer, J. B. Random walks on regular and irregular graphs. *SIAM J. Discret. Math.*, 1996. (page 11)

Cornish, R. Stochastic neural network symmetrisation in markov categories. *arXiv*, 2024.

Cotta, L., Yehuda, G., Schuster, A., and Maddison, C. J. Probabilistic invariant learning with randomized linear classifiers. In *NeurIPS*, 2023. (pages 2, 11)

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 1989.

Dasgupta, A., Kumar, R., and Sarlos, T. On estimating the average degree. In *WWW*, 2014. (page 11)

David, R. and Feige, U. Random walks with the minimum degree local rule have $o(n^2)$ cover time. *SIAM J. Comput.*, 2018. (pages 2, 3, 11)

de Luca, A. B. and Fountoulakis, K. Simulation of graph algorithms with looped transformers. *arXiv*, 2024. (page 11)

Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., and Ortega, P. A. Neural networks and the chomsky hierarchy. In *ICLR*, 2023. (page 11)

Devriendt, K. and Lambiotte, R. Discrete curvature on graphs from the effective resistance. *Journal of Physics: Complexity*, 2022. (page 11)

Ding, J., Lee, J. R., and Peres, Y. Cover times, blanket times, and majorizing measures. In *Annual ACM symposium on Theory of computing*, 2011. (page 11)

Doyle, P. G. and Snell, J. L. *Random walks and electric networks*. American Mathematical Soc., 1984. (page 11)

Dumitriu, I., Tetali, P., and Winkler, P. On playing golf with two balls. *SIAM J. Discret. Math.*, 2003. (pages 11, 32)

Elesedy, B. Group symmetry in pac learning. In *ICLR workshop on geometrical and topological representation learning*, 2022. (page 11)

Elesedy, B. *Symmetry and Generalisation in Machine Learning*. PhD thesis, University of Oxford, 2023. (page 11)

Falcon, W. Pytorch lightning. https://github.com/Lightning-AI/lightning, 2019. (page 12)

Fasino, D., Tonetto, A., and Tudisco, F. Hitting times for non-backtracking random walks. *arXiv*, 2021. (page 11)

Fatemi, B., Halcrow, J., and Perozzi, B. Talk like a graph: Encoding graphs for large language models. *arXiv*, 2023. (page 11)

Feige, U. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 1995. (page 11)

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. (page 12)

Fitzner, R. and van der Hofstad, R. Non-backtracking random walk. *Journal of Statistical Physics*, 2013. (page 15)

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

Georgakopoulos, A. and Winkler, P. New bounds for edge-cover by random walk. *Comb. Probab. Comput.*, 2014. (page 11)

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Giovanni, F. D., Rusch, T. K., Bronstein, M. M., Deac, A., Lackenby, M., Mishra, S., and Velickovic, P. How does over-squashing affect the power of gnns? *arXiv*, 2023. (pages 4, 11)

Giraldo, J. H., Skianis, K., Bouwmans, T., and Malliaros, F. D. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *CIKM*, 2023. (pages 1, 4, 5, 11)

Grady, L. J. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2006. (pages 11, 14)

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *KDD*, 2016. (pages 1, 3, 11, 15, 30)

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: decoding-enhanced bert with disentangled attention. In *ICLR*, 2021. (pages 1, 3, 12)

He, X., Bresson, X., Laurent, T., Perold, A., LeCun, Y., and Hooi, B. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *ICLR*, 2023. (pages 5, 14)

Hein, J. Wald's identity. (page 35)

Hornik, K., Stinchcombe, M. B., and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989. (page 4)

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. (page 41)

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020. (pages 5, 14)

Huang, Q., He, H., Singh, A., Lim, S.-N., and Benson, A. R. Combining label propagation and simple models out-performs graph neural networks. *arXiv*, 2020. (pages 5, 14)

Ikeda, S., Kubo, I., and Yamashita, M. The hitting and cover times of random walks on finite graphs using local degree information. *Theor. Comput. Sci.*, 2009. (page 11)

Ivanov, S. and Burnaev, E. Anonymous walk embeddings. In *ICML*, 2018.

Janson, S. and Peres, Y. Hitting times for random walks with restarts. *SIAM J. Discret. Math.*, 2012. (pages 11, 32)

Kahn, J. D., Linial, N., Nisan, N., and Saks, M. E. On the cover time of random walks on graphs. *Journal of Theoretical Probability*, 1989. (page 11)

Kallenberg, O. et al. *Probabilistic symmetries and invariance principles*. Springer, 2005.

Kempton, M. Non-backtracking random walks and a weighted ihara's theorem. *arXiv*, 2016. (page 11)

Kim, J., Nguyen, D., Min, S., Cho, S., Lee, M., Lee, H., and Hong, S. Pure transformers are powerful graph learners. In *NeurIPS*, 2022. (page 11)

Kim, J., Nguyen, D., Suleymanzade, A., An, H., and Hong, S. Learning probabilistic symmetrization for architecture agnostic equivariance. In *NeurIPS*, 2023. (pages 11, 41)

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Klubicka, F., Maldonado, A., Mahalunkar, A., and Kelleher, J. D. Synthetic, yet natural: Properties of wordnet random walk corpora and the impact of rare words on embedding performance. In *GWC*, 2019. (page 41)

Klubicka, F., Maldonado, A., Mahalunkar, A., and Kelleher, J. D. English wordnet random walk pseudo-corpora. In *LREC*, 2020. (page 41)

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023. (page 12)

Li, G., Müller, M., Ghanem, B., and Koltun, V. Training graph neural networks with 1000 layers. In *ICML*, 2021. (pages 5, 14)

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019. (page 12)

Loukas, A. What graph neural networks cannot learn: depth vs width. In *ICLR*, 2020. (page 11)

Lovász, L. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 1993. (page 11)

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Frozen pretrained transformers as universal computation engines. In *AAAI*, 2022. (page 3)

Lyle, C., van der Wilk, M., Kwiatkowska, M., Gal, Y., and Bloem-Reddy, B. On the benefits of invariance in neural networks. *arXiv*, 2020. (pages 2, 11)

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *NeurIPS*, 2019. (pages 5, 12)

Martinkus, K., Papp, P. A., Schesch, B., and Wattenhofer, R. Agent-based graph neural networks. In *ICLR*, 2023. (pages 5, 11, 12, 13)

McNew, N. Random walks with restarts, 3 examples, 2013. (pages 3, 11, 32, 33)

meta llama. llama3: The official meta llama 3 github site. `https://github.com/meta-llama`, 2024. (pages 1, 5, 12, 13, 14)

Micali, S. and Zhu, Z. A. Reconstructing markov processes from independent and anonymous experiments. *Discret. Appl. Math.*, 2016. (pages 3, 11)

Mitton, J. and Murray-Smith, R. Subgraph permutation equivariant networks. *arXiv*, 2021. (page 12)

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

Murphy, R. L., Srinivasan, B., Rao, V. A., and Ribeiro, B. Relational pooling for graph representations. In *ICML*, 2019. (pages 11, 12)

Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S. J., and Nguyen, T. M. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *ICML*, 2023. (pages 4, 5, 11)

Oliveira, R. Mixing and hitting times for finite markov chains, 2012. (page 11)

Ollivier, Y. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 2009. (page 11)

Page, L., Brin, S., Motwani, R., Winograd, T., et al. The pagerank citation ranking: Bringing order to the web, 1999. (pages 3, 11)

Panotopoulou, A. Bounds for edge-cover by random walks, 2013. (page 11)

Papp, P. A., Martinkus, K., Faber, L., and Wattenhofer, R. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *NeurIPS*, 2021. (pages 11, 12, 13)

Park, S., Ryu, N., Kim, G., Woo, D., Yun, S.-Y., and Ahn, S. Non-backtracking graph neural networks. In *NeurIPS Workshop: New Frontiers in Graph Learning*, 2023. (page 11)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. (page 12)

Peres, Y. and Sousi, P. Mixing times are hitting times of large sets. *Journal of Theoretical Probability*, 2015. (page 11)

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: online learning of social representations. In *KDD*, 2014. (pages 1, 2, 11)

Puny, O., Ben-Hamu, H., and Lipman, Y. Global attention improves graph networks generalization. *arXiv*, 2020. (page 11)

Rappaport, B., Gamage, A., and Aeron, S. Faster clustering via non-backtracking random walks. *arXiv*, 2017. (page 15)

Rothermel, D., Li, M., Rocktäschel, T., and Foerster, J. Don't sweep your learning rate under the rug: A closer look at cross-modal transfer of pretrained transformers. *arXiv*, 2021.

Sanford, C., Fatemi, B., Hall, E., Tsitsulin, A., Kazemi, M., Halcrow, J., Perozzi, B., and Mirrokni, V. Understanding transformer reasoning capabilities via graph algorithms. *arXiv*, 2024. (page 11)

Sato, R. Training-free graph neural networks and the power of labels as features. *arXiv*, 2024. (page 14)

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *SDM*, 2021. (pages 12, 13)

Sieradzki, Y., Hodos, N., Yehuda, G., and Schuster, A. Coin flipping neural networks. In *ICML*, 2022. (page 11)

Spielman, D. Spectral and algebraic graph theory. *Yale lecture notes, draft of December*, 2019. (page 11)

Tahmasebi, B., Lim, D., and Jegelka, S. The power of recursion in graph neural networks for counting substructures. In *AISTATS*, 2023.

Tan, Y., Zhou, Z., Lv, H., Liu, W., and Yang, C. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. In *NeurIPS*, 2023. (page 11)

Tönshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research*, 2023. (pages 1, 5, 11, 12, 13)

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*, 2022. (pages 1, 4, 5, 11)

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *arXiv*, 2023. (page 3)

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? In *NeurIPS*, 2023. (page 11)

Weiss, G., Goldberg, Y., and Yahav, E. Thinking like transformers. In Meila, M. and Zhang, T. (eds.), *ICML*, 2021. (page 11)

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. Huggingface's transformers: State-of-the-art natural language processing. *arXiv*, 2019. (page 12)

Wu, X., Ajorlou, A., Wu, Z., and Jadbabaie, A. Demystifying oversmoothing in attention-based graph neural networks. In *NeurIPS*, 2023. (page 4)

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019. (pages 1, 5, 12)

Ye, R., Zhang, C., Wang, R., Xu, S., and Zhang, Y. Language is all a graph needs. In *Findings of EACL*, 2024. (page 11)

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In *ICLR*, 2020. (pages 3, 4)

Zhang, B., Gai, J., Du, Y., Ye, Q., He, D., and Wang, L. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness. *arXiv*, 2024. (page 13)

Zhao, J., Qu, M., Li, C., Yan, H., Liu, Q., Li, R., Xie, X., and Tang, J. Learning on large-scale text-attributed graphs via variational inference. *arXiv*, 2022a. (pages 5, 14)

Zhao, J., Zhuo, L., Shen, Y., Qu, M., Liu, K., Bronstein, M. M., Zhu, Z., and Tang, J. Graphtext: Graph reasoning in text space. *arXiv*, 2023. (page 11)

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. *arXiv*, 2019. (page 4)

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *ICLR*, 2022b. (pages 5, 12, 13)

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. *ProQuest number: information to all users*, 2002. (pages 11, 14)

Zhu, X. J. Semi-supervised learning literature survey, 2005. (pages 11, 14)

Zuckerman, D. On the time to traverse all edges of a graph. *Inf. Process. Lett.*, 1991. (pages 3, 4, 11, 33, 37)

A. Appendix

A.1. Related Work

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 method (Section 2) is related to 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 in case of infinite graphs (Dumitriu et al., 2003; McNew, 2013; Janson & Peres, 2012). Random walks and their statistical properties are closely related to 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). This has motivated our analysis in Section 3.2, where we transfer the prior results on over-smoothing and over-squashing of message passing based on these properties (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) into the results on our approach. 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). While we adopt their techniques to make our walks and their records well-behaved, the difference is that we use a deep neural network to process the records and directly make predictions.

Random walks and graph learning In graph learning, random walks have received interest due to their scalability and compatibility to sequence learning methods. DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) use shallow skip-gram models on random walks to learn vertex embeddings. While CRaWl (Tönshoff et al., 2023) is the most relevant to our work, it uses convolutions on walks recorded in a different way from ours. Hence, its expressive power is controlled by the kernel size and does not always support universality. AgentNet (Martinkus et al., 2023) learns agents that walk on a graph while recurrently updating their features. While optimizing the walk strategy, this method may trade off efficiency as training requires recurrent roll-out of the entire network. Our method allows pairing simple and fast walkers with parallelizable networks such as transformers, which suffices for universality. WalkLM (Tan et al., 2023) proposed a fine-tuning method for language models on random walks on text-attributed graphs. While in a similar spirit, our result provides more principles as well as extending to purely topological graphs and training-free setups. 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, which we further discuss in Section A.2.3.

Probabilistic invariant neural networks Whenever a learning problem is compatible with symmetry, incorporating the associated invariance structure to the hypothesis class often leads to generalization benefit (Bronstein et al., 2021; Elesedy, 2022; 2023). This is also the case for probabilistic invariant neural networks (Lyle et al., 2020; Bloem-Reddy & Teh, 2020), which includes our approach. Probabilistic invariant networks have recently gained interest due to their potential of achieving higher expressive powers compared to deterministic counterparts (Cotta et al., 2023; Sieradzki et al., 2022). In graph learning, this is often achieved with stochastic symmetry breaking between vertices using randomized features (Loukas, 2020; Puny et al., 2020; Abboud et al., 2021; Kim et al., 2022), vertex orderings (Murphy et al., 2019; Kim et al., 2023), or dropout (Papp et al., 2021). Our approach can be understood as using random walk as a symmetry-breaking mechanism for probabilistic invariance, which provides an additional benefit of natural compatibility with sequence learning methods.

Language models 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., 2023b; 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.

Table 3: Graph separation results. We report accuracy of classifying training data rounded to the first decimal point. *, †, ‡, §, and ◇ are from Papp et al. (2021), Murphy et al. (2019), Zhao et al. (2022b), Martinkus et al. (2023), and Alvarez-Gonzalez et al. (2024), respectively. We trained AgentNet and CRaWl on SR25 with their reported best configurations on SR16 and CSL, respectively.

Method	CSL 1, 2-WL fails	SR16 3-WL fails	SR25 3-WL fails
GIN (Xu et al., 2019)	10.0%†	50.0%§	6.7%‡
PPGN (Maron et al., 2019)	100.0%§	50.0%§	6.7%‡
GIN-AK+ (Zhao et al., 2022b)	-	-	6.7%‡
PPGN-AK+ (Zhao et al., 2022b)	-	-	100.0%‡
GIN+ELENE (Alvarez-Gonzalez et al., 2024)	-	-	6.7%◇
GIN+ELENE-L (ED) (Alvarez-Gonzalez et al., 2024)	-	-	100.0%◇
GIN-RNI (Abboud et al., 2021; Sato et al., 2021)	16.0%*	-	-
GIN-RP (Murphy et al., 2019)	37.6%†	-	-
GIN-Dropout (Papp et al., 2021)	82.0%*	-	-
Random Walk AgentNet (Martinkus et al., 2023)	100.0%§	50.5%§	-
AgentNet (Martinkus et al., 2023)	100.0%§	100.0%§	6.7%
CRaWl (Tönshoff et al., 2023)	100.0%§	100.0%§	46.6%
DeBERTa-walk (Ours)	100.0%	100.0%	100.0%

A.2. Full Experiments

We perform a series of experiments to demonstrate random walk neural networks. We implement random walk algorithms in C++ based on Chenebaux (2020), which produces good throughput even without GPU acceleration (Tönshoff et al., 2023). Likewise, we implement recording functions in C++ which perform anonymization with named neighborhoods and produce plain text. Pseudocode of our implementation is given in Appendix A.4. We choose reader neural networks as pre-trained language models, with a prioritization on long inputs in accordance to Section 3.1. For fine-tuning experiments, we use DeBERTa-base (He et al., 2021) that supports up to 12,276 tokens, and for training-free experiments, we use instruction-tuned Llama 3 (meta llama, 2024) that support up to 8,192 tokens. We implement our pipelines in PyTorch (Paszke et al., 2019) with libraries (Fey & Lenssen, 2019; Falcon, 2019; Wolf et al., 2019; Kwon et al., 2023) that support multi-GPU training and inference.

A.2.1. Graph Separation

We first verify the claims on expressive power in Section 3.1 using three synthetic datasets where the task is recognizing the isomorphism type of an input graph. This is framed as fitting an N-way classification problem on N non-isomorphic graphs. The graphs are chosen such that a certain WL graph isomorphism test fails, and so does their related family of message passing neural networks.

The Circular Skip Links (CSL) graphs dataset (Murphy et al., 2019) contains 10 non-isomorphic regular graphs with 41 vertices of degree 4. Distinguishing these graphs requires computing lengths of skip links (Martinkus et al., 2023; Chen et al., 2019), and 1-WL and 2-WL tests fail. The 4×4 rook's and Shrikhande graphs (Alvarez-Gonzalez et al., 2024), which we call SR16, are a pair of strongly regular graphs with 16 vertices of degree 6. Distinguishing the pair requires detecting 4-cliques (Martinkus et al., 2023), and 3-WL test fails (Alvarez-Gonzalez et al., 2024). The SR25 dataset (Balcilar et al., 2021) contains 15 strongly regular graphs with 25 vertices of degree 12, on which 3-WL test fails (Alvarez-Gonzalez et al., 2024). SR25 is challenging and has only been solved by a handful of carefully designed subgraph-based message passing algorithms (Mitton & Murray-Smith, 2021; Zhao et al., 2022b; Alvarez-Gonzalez et al., 2024). Examples of the considered graphs and random walks on them can be found in Appendix A.4.

Our random walk neural network uses the minimum degree local rule from Equation (6) with non-backtracking (Algorithm 1), and recording function with anonymization and named neighborhoods that produces plain text (Algorithm 3). We use pre-trained DeBERTa-base language model as the reader neural network, and fine-tune it with cross-entropy loss for at most 100k steps using AdamW optimizer (Loshchilov & Hutter, 2019) with 2e-5 learning rate and 0.01 weight decay on on 8× RTX 3090 GPUs. While the model supports 12,276 tokens in maximum, we use 512 tokens which is the maximum allowed by memory constraint. We use batch size 256, and accumulate gradients for 8 steps for SR25. At test time, we ensemble 4

Table 4: Substructure (8-cycle) counting results. We report normalized mean absolute error (MAE) on test set 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 (Zhang et al., 2024)	.0917	.1156
Subgraph GNN (Zhang et al., 2024)	.0515	.0715
Local 2-GNN (Zhang et al., 2024)	.0433	.0478
Local 2-FGNN (Zhang et al., 2024)	.0422	.0438
Local 2-FGNN (large)	.0932	.1121
MP-RNI (Abboud et al., 2021; Sato et al., 2021)	.3610	.3650
MP-Dropout (Papp et al., 2021)	.1359	.1393
AgentNet (Martinkus et al., 2023)	.1107	N/A
CRaWl (Tönshoff et al., 2023)	.0526	.0725
DeBERTa-walk (Ours)	.0459	.0465

stochastic predictions of the network by averaging classification logits.

The results are in Table 3. Message passing neural networks that align with certain WL test fail when asked to solve harder problems, e.g. GIN aligns with 1-WL and fails in CSL. Recent algorithms that use subgraphs and structural features show improved performance, with state-of-the-art such as ELENE-L (ED) solving SR25. Alternative approaches based on stochastic symmetry-breaking e.g. random node identification, often fail on CSL although universal approximation is possible in theory (Abboud et al., 2021), possibly due to learning difficulties. For algorithms based on walks, AgentNet and CRaWl solve CSL and SR16, while failing on SR25. This is because learning a policy to walk in AgentNet can be challenging in complex tasks especially at the early stage of training, and the expressiveness of CRaWl is limited by the receptive field of convolution layers. Our approach based on DeBERTa language model overcomes the problems, being the first walk-based algorithm that solves SR25.

In Appendix A.5, we further provide visualizations of learned attentions by mapping attention weights on text records of walks to input graphs. We find that the models often focus on sparse, connected substructures, which presumably provide discriminative information on the isomorphism types.

A.2.2. Substructure Counting

We test our model on more 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 are known to require a particularly high expressive power (Zhang et al., 2024). We experiment with both graph-level and vertex-level counting i.e. counting 8-cycles containing the queried vertex, following Zhang et al. (2024). We use the data splits from the previous works (Chen et al., 2020; Zhao et al., 2022b; Zhang et al., 2024), while excluding disconnected graphs following our modeling assumption in Section 2.1. Our random walk neural network uses the same design to Section A.2.1, and we train them with L1 loss for at most 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 stochastic predictions by averaging for graph- and vertex-level tasks, respectively.

The results are in Table 4, and overall in line with Section A.2.1. While simple message passing shows 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, message passing 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.

A.2.3. In-Context Transductive Classification

Previous results considered synthetic tasks on undirected, unattributed, and relatively small graphs. We now show that our approach based on frozen Llama 3 (meta llama, 2024) language models can solve real-world problem on a large graph

Table 5: Transductive classification test accuracy on ogbn-arxiv. † denotes using validation labels for in-context learning or label propagation, as done in Huang et al. (2020). For Llama 3, we use instruction-tuned models, and ensemble 5 predictions by voting for both one-shot learning and our algorithm.

Method	Accuracy
MLP (Hu et al., 2020)	55.50%
node2vec (Hu et al., 2020)	70.07%
GCN (Hu et al., 2020)	71.74%
GAT (Li et al., 2021)	73.91%
RevGAT (Li et al., 2021)	74.26%
Label propagation (Huang et al., 2020)	68.50%
C&S (Huang et al., 2020)	72.62%
C&S† (Huang et al., 2020)	74.02%
Llama2-13b zero-shot (He et al., 2023)	44.23%
GPT-3.5 zero-shot (He et al., 2023)	73.50%
DeBERTa, fine-tuned (Zhao et al., 2022a)	74.13%
Llama3-8b zero-shot	50.20%
Llama3-8b one-shot	52.49%
Llama3-8b one-shot†	52.54%
Llama3-8b-walk (Ours)	71.10%
Llama3-8b-walk (Ours)†	73.11%
Llama3-70b zero-shot	65.33%
Llama3-70b one-shot†	67.57%
Llama3-70b-walk (Ours)†	**74.75%**

with directed edges and textual attributes, without any training. We use ogbn-arxiv (Hu et al., 2020), a citation network of 169,343 arXiv papers with title and abstract attributes. The task is transductive classification into 40 areas such as `"cs.AI"` using a subset of labeled vertices.

We consider two representative types of baselines. The first reads title and abstract of each vertex using a language model and solves vertex-wise classification problem, ignoring graph structure. The second initializes vertex features as language model embeddings and trains a graph neural network, at the risk of over-compressing the text. Our algorithm allows taking the best of both worlds. We design the recording function so that, not only it does basic operations such as anonymization, it records a complete information of the local subgraph including title and abstract, edge directions, and notably, **labels in case of labeled vertices** (Appendix A.4) (Sato, 2024). The resulting record naturally includes a number of input-label pairs of the classification task at hand, implying that we can frame transductive classification problem as a simple in-context learning problem (Brown et al., 2020). This allows us to use a frozen Llama 3 (meta llama, 2024) language model directly on transductive classification without any training.

The results are in Table 5. Notably, our model based on frozen Llama 3 is able to outperform graph neural networks on text embeddings, as well as language models that perform vertex-wise predictions ignoring graph structures such as GPT-3.5 and fine-tuned DeBERTa. We especially note that our model largely outperforms one-shot baselines, which are given 40 randomly chosen labeled examples (one per class). This is surprising as our model observes fewer vertices, 29.17 in average, due to other recorded information. This is presumably since the record produced by our algorithm faithfully encodes graph structure, providing more useful input for the language model to quickly learn the task in-context compared to randomly chosen shots. Overall, the results show that our approach allows a pure language model to understand and leverage graph structures even without training. In Appendix A.5, we provide a visualization of the internal attention weights, verifying that the model makes use of the graph structure recorded by the random walk to make predictions.

As a final note, our approach is related to label propagation algorithms (Zhu & Ghahramani, 2002; Zhu, 2005; Grady, 2006) for transductive classification, which makes predictions by running random walks and probing the distribution of visited labeled vertices. The difference is that, in our approach, the reader neural network i.e. language model can appropriately use other information such as attributes, as well as do meaningful non-linear processing rather than simply probing the input. As shown in Table 5, our approach outperforms label propagation, verifying our intuition. This result also supports the proposal of Sato (2024) of using labels as features for training-free neural networks for transductive classification on graphs.

A.3. Second-Order Random Walks (Section 2.1)

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 $v_{t+1} \neq v_{t-1}$:

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

where $\mathrm{Prob}[v_{t+1} = x | v_t = j]$ 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:

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

where the probability $\mathrm{Prob}[v_{t+1} = x | v_t = j]$ 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 $d(v_{t-1}, v_{t+1})$:

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

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 Theorem 2.2 to the above second-order random walks:

Theorem 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.6.3. □

A.4. Main Algorithm

We outline the main algorithms for the random walk and the recording function described in Section 2 and used in Section 4. 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.3 and the minimum degree local rule conductance $c_G(\cdot)$ given in Equation (6) by default.

Based on the algorithms, in Figures 2, 3, and 4 we illustrate the task formats used for graph separation experiments in Section A.2.1 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 5, 6, and 7 we show task formats for transductive classification on arXiv citation network in Section A.2.3.

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₊ */
for (u, x) ∈ E(G) do
    p(u, x) ← c_G(u, x) / Σ_{y∈N(u)} c_G(u, y)                    // Equation (4)
/* starting vertex v_0 */
if v is given then
    /* query starting vertex */
    v_0 ← v
else
    /* sample starting vertex */
    v_0 ~ Uniform(V(G))
/* random walk v_1 → ··· → v_l, restart flags r_1, ..., r_l */
v_1 ~ Categorical({p(v_0, x) : x ∈ N(v_0)})
r_1 ← 0
for t ← 2 to l do
    /* restart flag r_t */
    r_t ← 0
    if α is given then
        /* if restarted at t − 1, do not restart */
        if r_{t−1} = 0 then
            r_t ~ Bernoulli(α)
    else if k is given then
        if t ≡ 0 (mod k) then
            r_t ← 1
    /* random walk v_t */
    if r_t = 0 then
        S ← N(v_{t−1}) \ {v_{t−2}}
        if S ≠ ∅ then
            /* non-backtracking */
            v_t ~ Categorical({p(v_{t−1}, x) / Σ_{y∈S} p(v_{t−1}, y) : x ∈ S})    // Equation (13)
        else
            /* begrudgingly backtracking */
            v_t ← v_{t−2}
    else
        /* restart */
        v_t ← 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(·) */
S ← {v_0}
id(v_0) ← 1
/* text record z */
z ← id(v_0)
for t ← 1 to l do
    /* anonymization v_t ↦ id(v_t) */
    if v_t ∉ S then
        S ← S ∪ {v_t}
        id(v_t) ← |S|
    /* text record z */
    if r_t = 0 then
        /* record walk v_{t-1} → v_t */
        z ← z + "-" + id(v_t)
    else
        /* record restart v_t = v_0 */
        z ← z + ";" + id(v_t)
```

Algorithm 3: Recording function, using anonymization and named neighborhoods

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

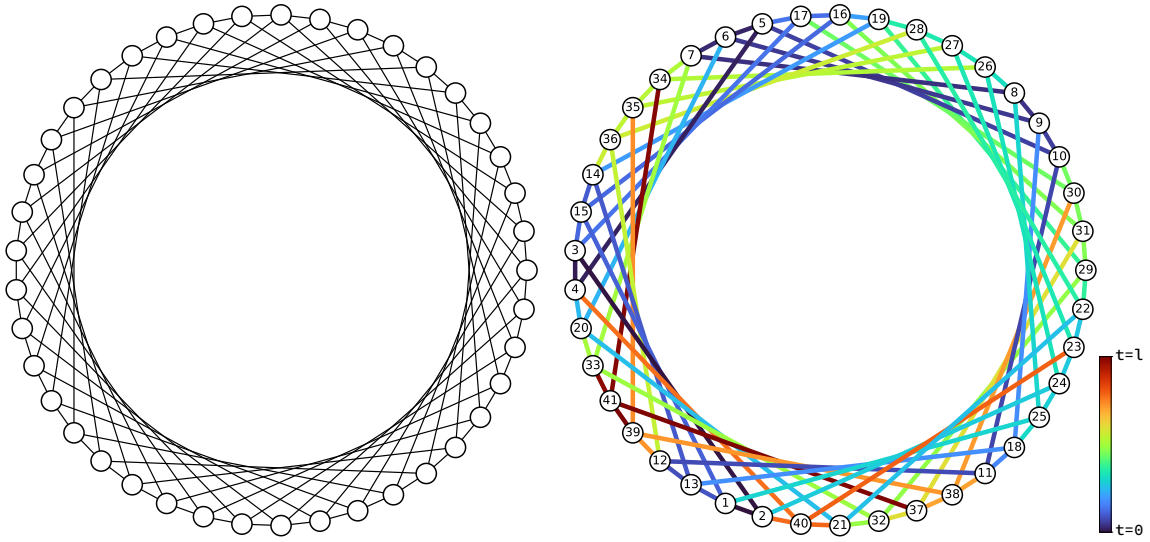
```
/* named vertices S, namespace id(·), recorded edges T */
S ← {v_0}
id(v_0) ← 1
T ← ∅
/* text record z */
z ← id(v_0)
for t ← 1 to l do
    /* anonymization v_t ↦ id(v_t) */
    if v_t ∉ S then
        S ← S ∪ {v_t}
        id(v_t) ← |S|
    /* text record z */
    if r_t = 0 then
        /* record walk v_{t-1} → v_t */
        z ← z + "-" + id(v_t)
        T ← T ∪ {(v_{t-1}, v_t), (v_t, v_{t-1})}
        /* named neighborhood U */
        U ← N(v_t) ∩ S
        if U ≠ ∅ then
            /* sort in ascending order id(u_1) < ··· < id(u_{|U|}) */
            [u_1, ..., u_{|U|}] ← SortByKey(U, id)
            for u ← u_1 to u_{|U|} do
                /* record named neighbors v_t → u */
                if (v_t, u) ∉ T then
                    z ← z + "#" + id(u)
                    T ← T ∪ {(v_t, u), (u, v_t)}
    else
        /* record restart v_t = v_0 */
        z ← z + ";" + id(v_t)
```

Algorithm 4: Recording function for transductive classification on arXiv network (Section A.2.3)

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 $L \subset V(G)$

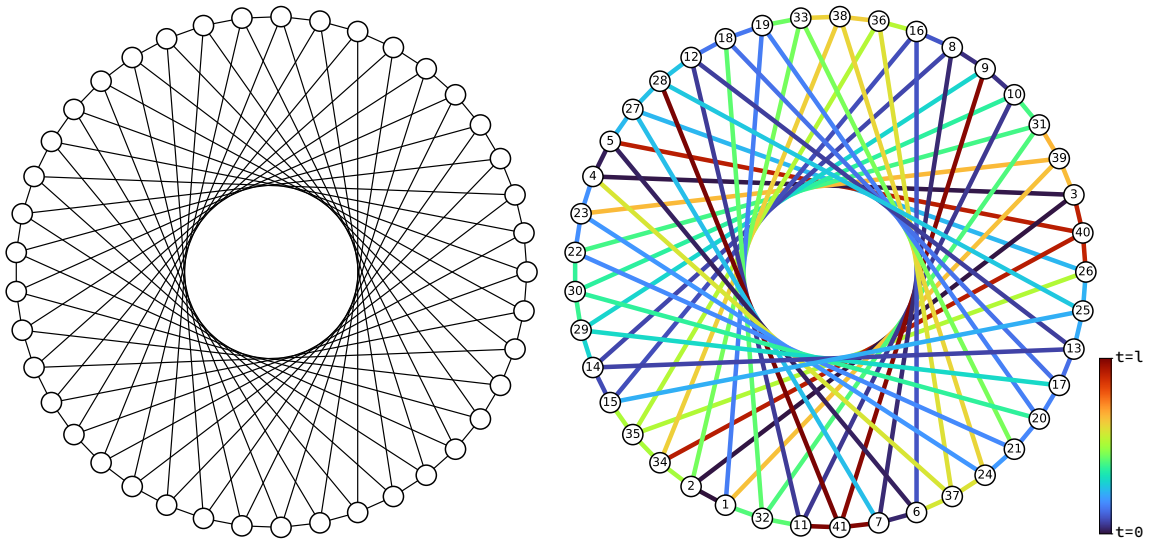
Result: Text record $\mathbf{z}$

/* named vertices S, namespace $\mathrm{id}(\cdot)$, recorded edges T */
$S \leftarrow \{v_0\}$
$\mathrm{id}(v_0) \leftarrow 1$
$T \leftarrow \emptyset$
/* text record $\mathbf{z}$ */
/* record starting vertex */
$\mathbf{z} \leftarrow$ "Paper 1 - Title: $\{\mathbf{t}_{v_0}\}$", Abstract: $\{\mathbf{a}_{v_0}\}$"
for $t \leftarrow 1$ **to** l **do**
  /* anonymization $v_t \mapsto \mathrm{id}(v_t)$ */
  if $v_t \notin S$ **then**
    $S \leftarrow S \cup \{v_t\}$
    $\mathrm{id}(v_t) \leftarrow |S|$
  /* text record $\mathbf{z}$ */
  if $r_t = 0$ **then**
    /* record walk $v_{t-1} \to v_t$ with direction */
    if $(v_{t-1}, v_t) \in E(G)$ **then**
      $\mathbf{z} \leftarrow \mathbf{z} +$ " Paper $\{\mathrm{id}(v_{t-1})\}$ cites Paper $\{\mathrm{id}(v_t)\}$"
    else
      $\mathbf{z} \leftarrow \mathbf{z} +$ " Paper $\{\mathrm{id}(v_{t-1})\}$ is cited by Paper $\{\mathrm{id}(v_t)\}$"
    $T \leftarrow T \cup \{(v_{t-1}, v_t), (v_t, v_{t-1})\}$
    /* record title $\mathbf{t}_v$, abstract $\mathbf{a}_v$, and label $\mathbf{y}_v$ if v is labeled */
    if $\mathrm{id}(v_t) = |S|$ **then**
      $\mathbf{z} \leftarrow \mathbf{z} +$ " - $\{\mathbf{t}_v\}$"
      if $v \in L$ **then**
        $\mathbf{z} \leftarrow \mathbf{z} +$ ", Category: $\{\mathbf{y}_v\}$"
      $\mathbf{z} \leftarrow \mathbf{z} +$ ", Abstract: $\{\mathbf{a}_v\}$"
    else
      $\mathbf{z} \leftarrow \mathbf{z} +$ "."
    /* named neighborhood U */
    $U \leftarrow N(v_t) \cap S$
    if $U \neq \emptyset$ **then**
      /* sort in ascending order $\mathrm{id}(u_1) < \cdots < \mathrm{id}(u_{|U|})$ */
      $[u_1, ..., u_{|U|}] \leftarrow \mathrm{SortByKey}(U, \mathrm{id})$
      for $u \leftarrow u_1$ **to** $u_{|U|}$ **do**
        /* record named neighbors $v_t \to u$ with directions */
        if $(v_t, u) \notin T$ **then**
          if $(v_t, u) \in E(G)$ **then**
            $\mathbf{z} \leftarrow \mathbf{z} +$ " Paper $\{\mathrm{id}(v_t)\}$ cites Paper $\{\mathrm{id}(u)\}$."
          else
            $\mathbf{z} \leftarrow \mathbf{z} +$ " Paper $\{\mathrm{id}(v_t)\}$ is cited by Paper $\{\mathrm{id}(u)\}$."
          $T \leftarrow T \cup \{(v_t, u), (u, v_t)\}$
  else
    /* record restart */
    $\mathbf{z} \leftarrow \mathbf{z} +$ " Restart at Paper 1."



Graph Random Walk

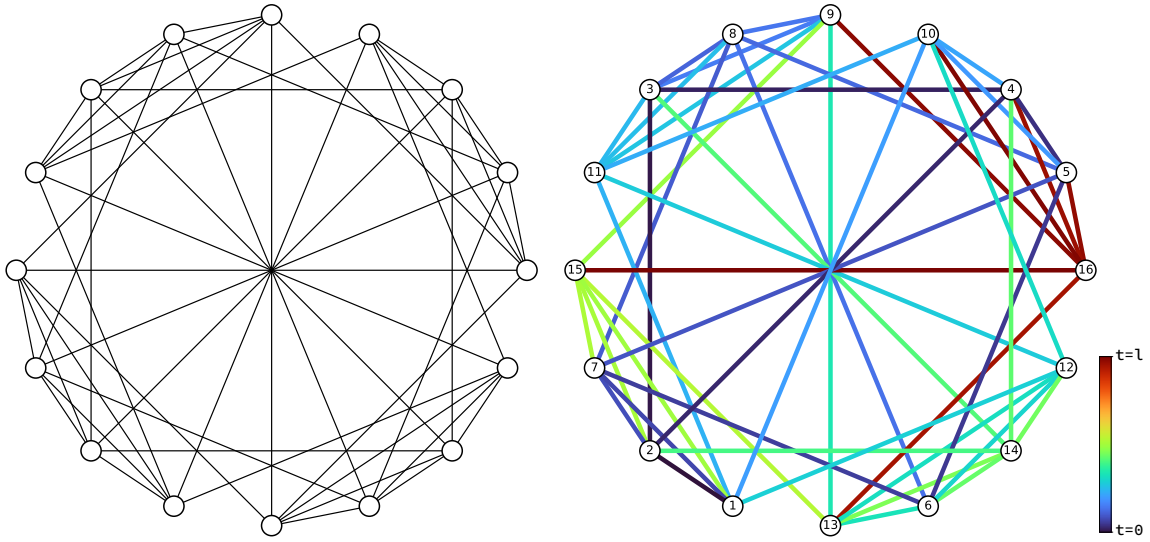
Record	[CLS]1-2-3-4-5-6-7-8-9#6-10#5-11-12-13#1-14-15#1#3-16-17#3#5-5-10-11-18#9#13-13-14-19#16-16-17-5-6-20#4-21-22-23-24#2-25#1#8#18-8-26#24-27#23-28#19#22-22-29#19-19-14-13-18-11-10-5-17-30#10-31#16#29-29-32#21-33#7#20-7-34#26-35#27-36#12#14#28-28-22-29-31-37#32-32-33-7-6-5-4-20-6-5-4-20-33-32-37-38#11#30-39#12#35-12-11-10-5-4-40#2#21#23-2-24-23-27-28-19-16-1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



Graph Random Walk

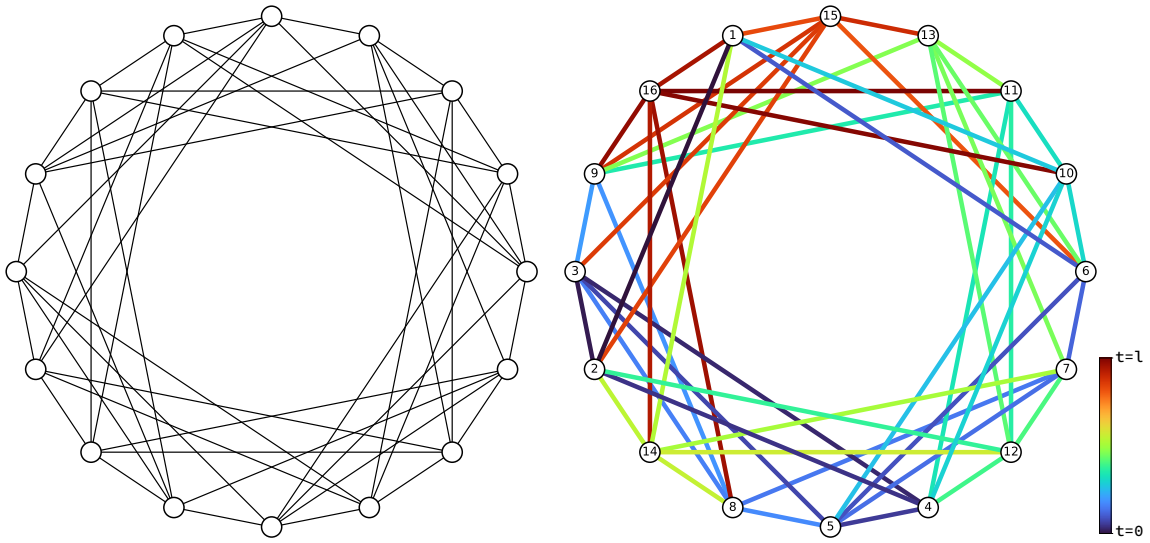
Record	[CLS]1-2-3-4-5-6-7-8-9-10-11-12-13-14#8-15-16#6#8-6-7-8-14-13-17-18#12-19#1-20#17-21-22-23#4-24#21-21-20-17-13-25#15-26-27#5#7-28#12#25-12-13-14-29#9#17-17-13-25-28-12-13-14-8-9-10-30#20#22#29-22-31#10-32#1#11#18-1-19-33#2#21-21-22-31-32-1-19-33-2-34-35#15#26-36#16-16-15-25-28-27-7-6-37#4#24#36-4-23-24-38#33#34#36-34-2-1-39#3#23#31-23-24-38-33-19-1-2-3-39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

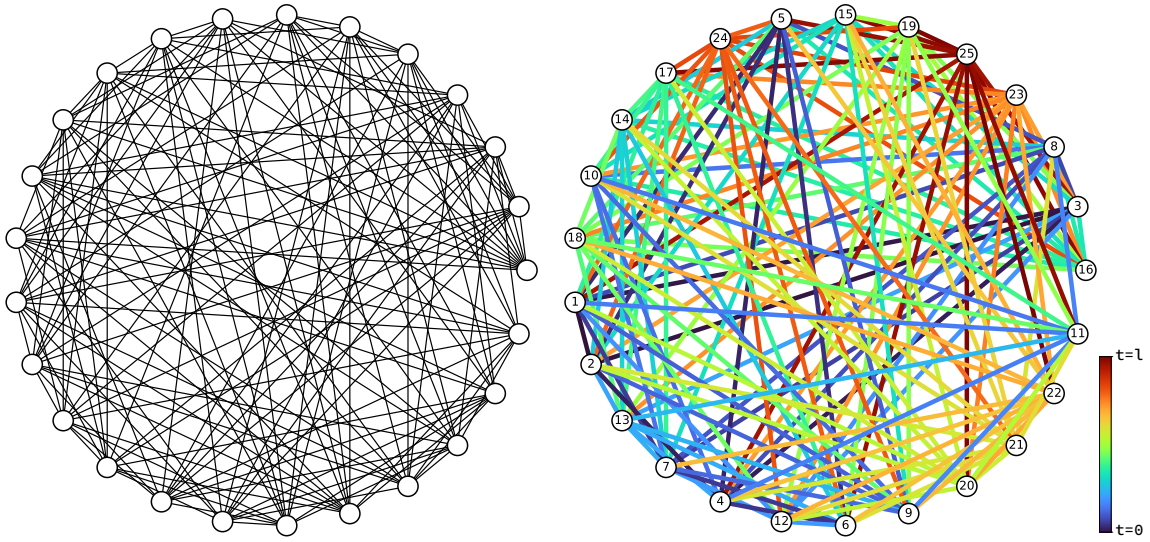
Record	[CLS]1-2-3-4#2-5-6-7#1#2#5-8#3#5#6-9#3-3-4-5-10#1#4-11#1#3#8#9-12#1#6#10-13#6# 9-6-5-8-7-2-14#3#4#6#12#13-13-9-15#1#2#7#13-7-6-13-12-1-2-4-10-12-11-1-7-15-1- 11-12-10-4-14-13-15-2-7-8-9-3-2-7-8-5-7-2-3-14-13-15-1-2-14-13-16#4#5#9#10#15-5- 10-1-11-3-14-4-16-10-1-12-13-6-12-13-6-12-10-11-12-10-11-1-7-6-13-15-16-5-4-14-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

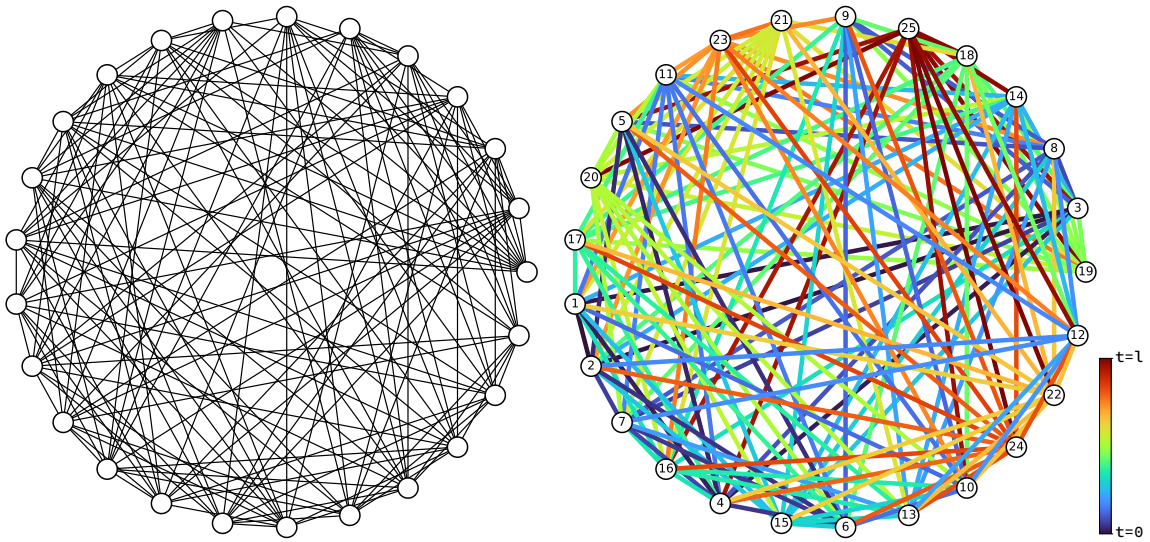
Record	[CLS]1-2-3-4#2-5#3-6#1-7#5-8#3#5-9#3-3-4-5-10#1#4#6-11#4#9-12#2#4#7-13#6#7#9#11- 7-14#1#2#8#12-8-5-6-7-13-9-11-4-5-3-4-12-14-1-10-6-15#1#2#3#9#13-2-14-16#1#8#9# 10#11-10-11-4-10-6-1-16-8-7-5-8-9-3-5-7-8-14-7-6-15-2-3-8-14-2-1-16-11-4-10-6- 15-13-6-1-16-10-6-1-16-14-12-13-7-12-4-3-2-12-11-4-5-7-13-15-3-5-6-1-15-9-13-11- 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 3: SR16 graphs and text records of random walks from Algorithms 1 and 3. The task is graph classification into two isomorphism types $\{4 \times 4$ rook's graph, Shrikhande graph$\}$. In random walks, we label vertices by anonymization and color edges by their time of discovery.



Graph | Random Walk

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



Graph | Random Walk

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 4: 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 5: 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 ex plain.
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 6: 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 7: 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.

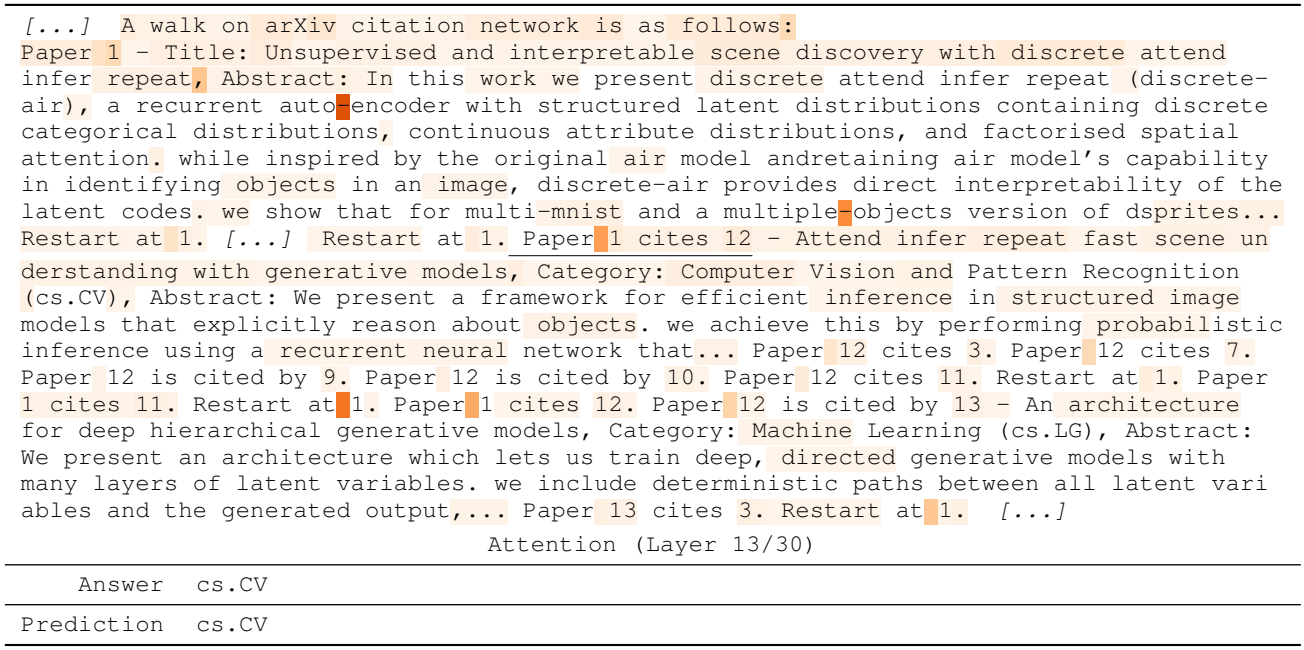


Figure 8: Example of attention in a frozen Llama 3 8B applied on arXiv transductive classification. Attention weights are averaged over all 30 heads.

A.5. Attention Visualizations

In Figure 8, we show an example of self-attention in the frozen Llama 3 8B model applied to arXiv transductive classification (Section A.2.3). 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 9, 10, and 11, we provide examples of self-attention in DeBERTa models trained for graph separation (Section A.2.1). We first show attention weights on text record of random walk from the `[CLS]` query token. Then, we further 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 $v_t \to v_{t+1}$, 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 9) 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 to infer isomorphism types of CSL graphs.

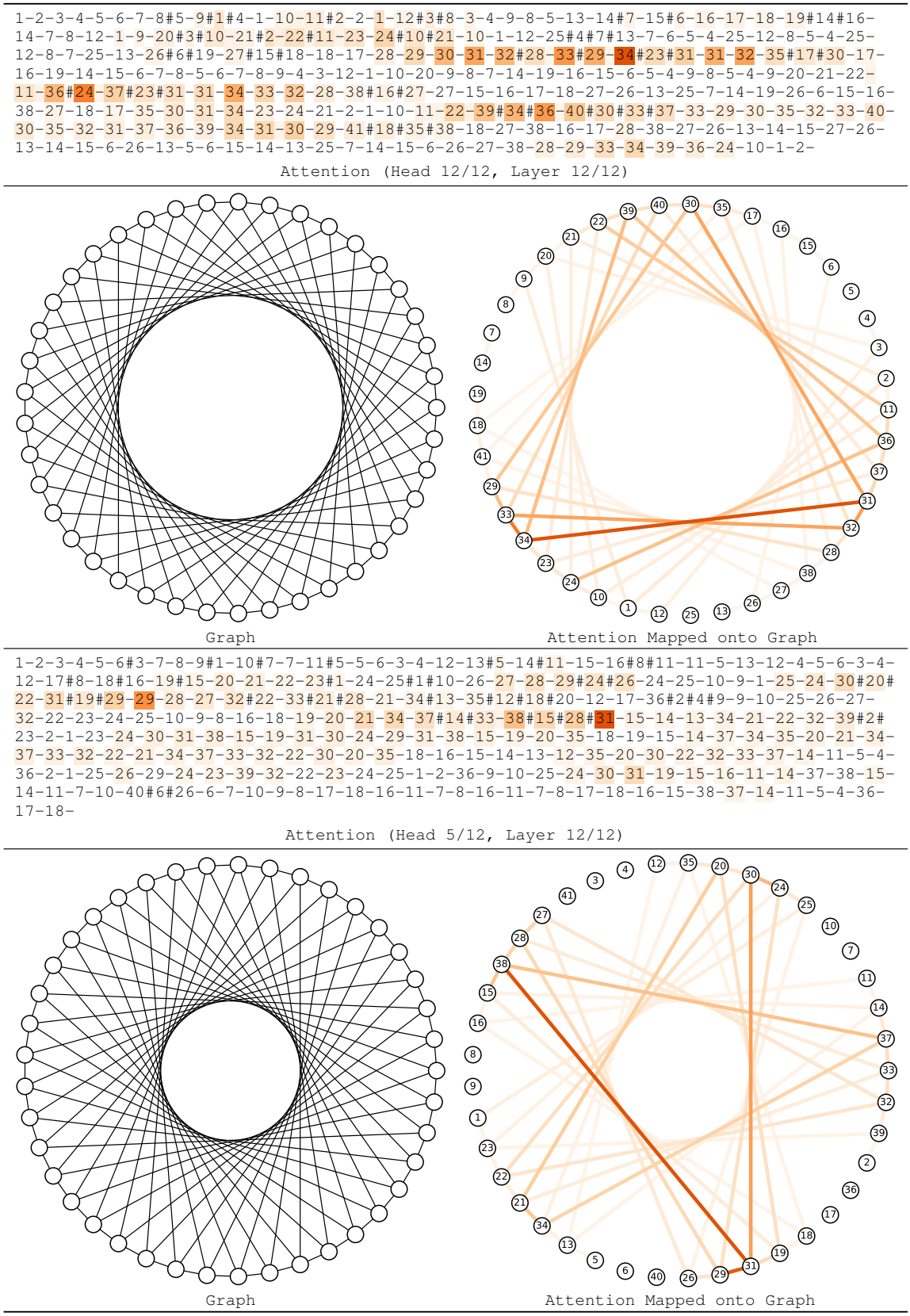


Figure 9: Examples of attention from [CLS] token in a DeBERTa trained on CSL graph separation, forming approximate cycles using skip links. This is presumably related to measuring the lengths of skip links, which provides sufficient information to infer isomorphism types of CSL graphs.

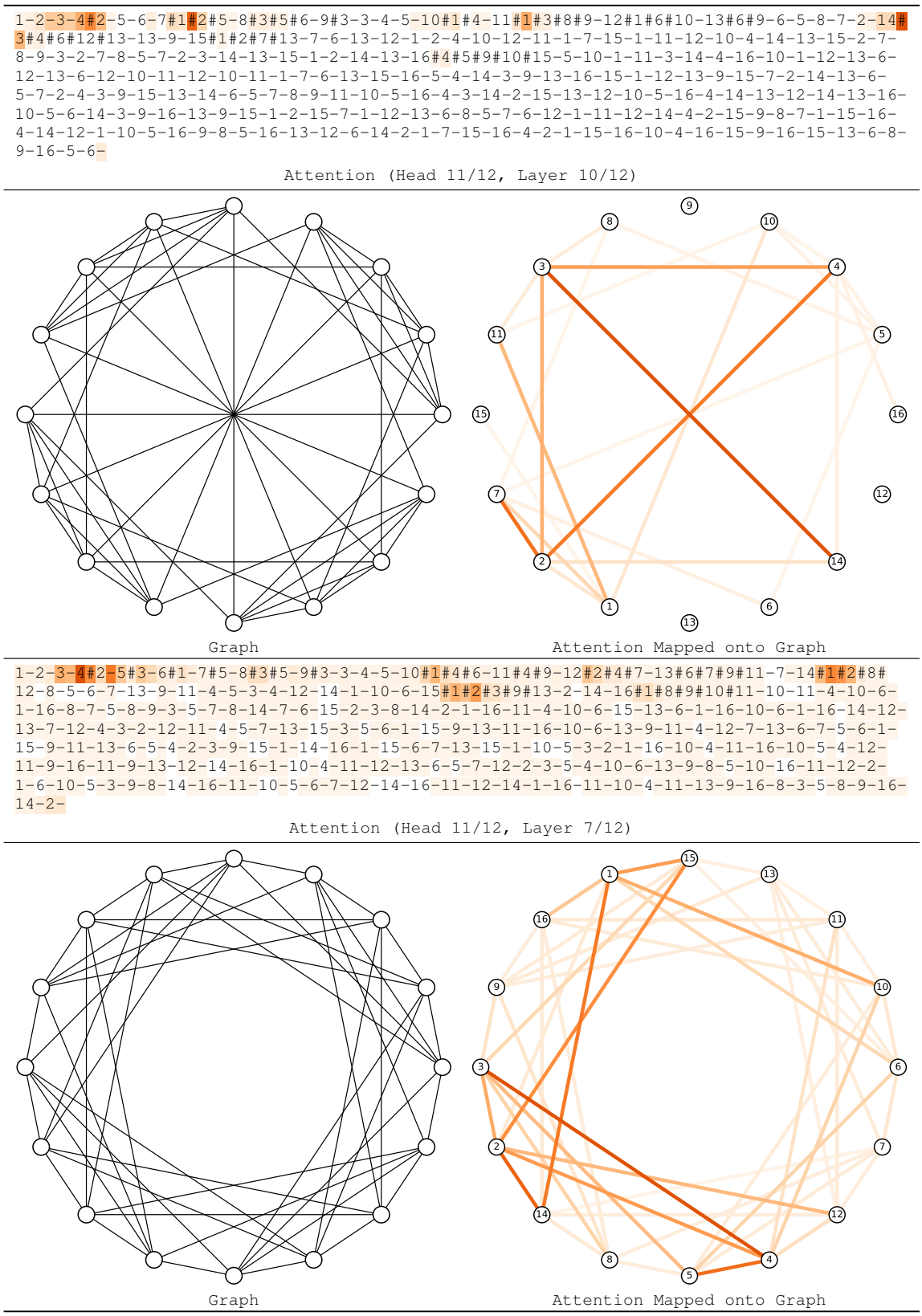


Figure 10: Examples of attention from [CLS] token in a DeBERTa trained on SR16 graph separation. The model tend to focus on neighborhood records that form a sparse connected substructure, which we conjecture to provide discriminative information on the isomorphism type of SR16 graphs.

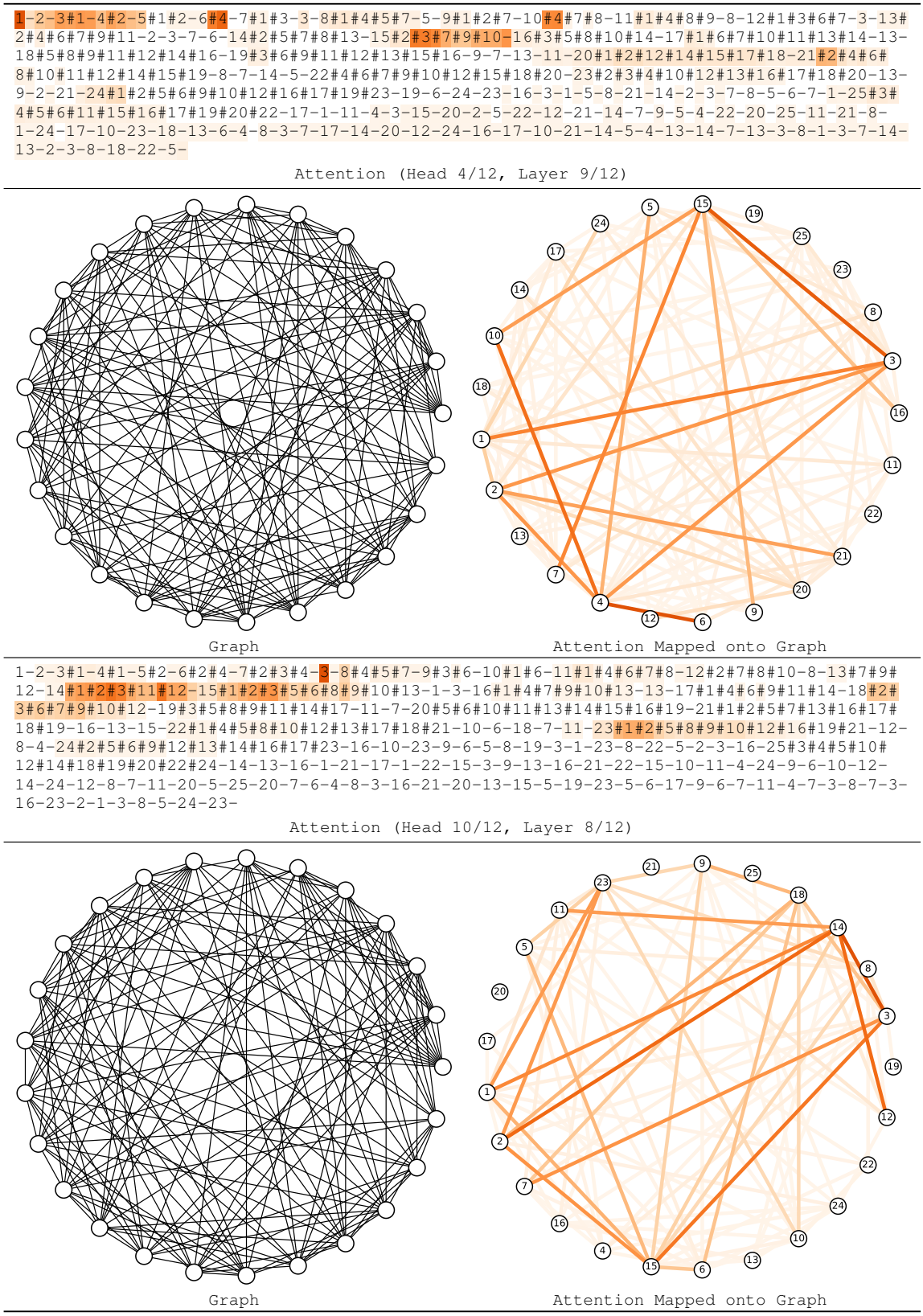


Figure 11: Examples of attention from `[CLS]` token in a DeBERTa trained on SR25 graph separation. The model tend to focus on neighborhood records that form a sparse connected substructure, which we conjecture to provide discriminative information on the isomorphism type of SR25 graphs.

A.6. 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 $G \overset{\pi}{\simeq} H$.

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 2.1) 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.6.5. 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:

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

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:

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

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:

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

A.6.1. Proof of Theorem 2.1 (Section 2.1)

Theorem 2.1. *$X_\theta(\cdot)$ is invariant if its random walk algorithm and recording function are invariant.*

Proof. We recall the invariance of a 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 a 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 (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. □

A.6.2. Proof of Theorem 2.2 (Section 2.1)

We first show a useful lemma.

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

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

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

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

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

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

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

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

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

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

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

This shows Equation (3) and completes the proof. □

We now prove Theorem 2.2.

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

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

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 $c_G : E(G) \to \mathbb{R}_+$:

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

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

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

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

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

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. □

A.6.3. Proof of Theorem A.1 (Appendix A.3)

We extend the proof of Theorem 2.2 given in Appendix A.6.2 to second-order random walks discussed in Appendix A.3. We first show a useful lemma:

Lemma A.3. *A second-order random walk algorithm is invariant 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 invariance of starting vertex v_0 and first transition $v_0 \to v_1$ are:

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

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

and the 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}):

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

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 the 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. □

We now prove Theorem A.1.

Theorem 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. 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 the 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 $\mathrm{Uniform}(V(G))$ (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. We rewrite Equation (36) as follows:

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

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.3). 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:

$$\begin{aligned}
&\mathrm{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] \\
&\qquad := \begin{cases} \dfrac{\mathrm{Prob}[v_t = x | v_{t-1} = j]}{\sum_{y \in N(j) \setminus \{i\}} \mathrm{Prob}[v_t = y | v_{t-1} = j]} & \text{for } x \neq i, \\ 0 & \text{for } x = i, \end{cases} \\
&\qquad = \begin{cases} \dfrac{\mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j)]}{\sum_{\pi(y) \in N(\pi(j)) \setminus \{\pi(i)\}} \mathrm{Prob}[u_t = \pi(y) | u_{t-1} = \pi(j)]} & \text{for } \pi(x) \neq \pi(i), \\ 0 & \text{for } \pi(x) = \pi(i), \end{cases} \\
&\qquad = \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \quad \forall G \overset{\pi}{\simeq} H. \qquad (41)
\end{aligned}$$

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. For node2vec random 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 $d(i, x) = d(\pi(i), \pi(x))$:

$$\begin{aligned}
\alpha(i, x) &:= \begin{cases} 1/p & \text{for } d(i, x) = 0, \\ 1 & \text{for } d(i, x) = 1, \\ 1/q & \text{for } d(i, x) = 2. \end{cases} \\
&= \begin{cases} 1/p & \text{for } d(\pi(i), \pi(x)) = 0, \\ 1 & \text{for } d(\pi(i), \pi(x)) = 1, \\ 1/q & \text{for } d(\pi(i), \pi(x)) = 2. \end{cases} \\
&= \alpha(\pi(i), \pi(x)). \qquad (42)
\end{aligned}$$

Then we can show Equation (40) by:

$$\begin{aligned}
\mathrm{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] &:= \frac{\alpha(i, x) \, \mathrm{Prob}[v_t = x | v_{t-1} = j]}{\sum_{y \in N(j)} \alpha(i, y) \, \mathrm{Prob}[v_t = y | v_{t-1} = j]}, \\
&= \frac{\alpha(\pi(i), \pi(x)) \, \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j)]}{\sum_{\pi(y) \in N(\pi(j))} \alpha(\pi(i), \pi(y)) \, \mathrm{Prob}[u_t = \pi(y) | u_{t-1} = \pi(j)]}, \\
&= \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \quad \forall G \overset{\pi}{\simeq} H. \qquad (43)
\end{aligned}$$

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

A.6.4. Proof of Theorem 2.3 (Section 2.1)

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

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

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:

$$\begin{aligned}
\mathrm{id}'(\pi(v_t)) &= 1 + \arg\min_i [\pi(v_i) = \pi(v_t)], \\
&= 1 + \arg\min_i [v_i = v_t], \\
&= \mathrm{id}(v_t), \quad \forall G \overset{\pi}{\simeq} H. \qquad (45)
\end{aligned}$$

The second equality is due to the fact that π is a bijection. This completes the proof for anonymization. For the named neighborhoods, 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 $\mathbf{z} \coloneqq q(v_0 \to \cdots \to v_{t-1}, G)$, and $T' \subseteq E(H)$ be the set of edges recorded by $\mathbf{z}' \coloneqq q(\pi(v_0) \to \cdots \to \pi(v_{t-1}), H)$. 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:

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

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

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

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

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

and consequently:

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

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 neighborhoods is invariant. □

A.6.5. 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 cover time and edge cover time 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 cover time and edge cover time cannot be finitely bounded. □

A.6.6. 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:

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

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:

$$\begin{aligned} H(u,x) &= \mathbb{E}[|w|_x | w \in W(u,x)], \\ &= \int_{W(u,x)} |w|_x \, dP(w | w \in W(u,x)). \end{aligned} \tag{60}$$

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:

$$\begin{aligned} H(u,x) &= \int_{W(u,v,x) \cup W(u,v^-,x)} |w|_x \, dP(w|W(u,x)), \\ &= \int_{W(u,v,x)} |w|_x \, dP(w|W(u,x)) + \int_{W(u,v^-,x)} |w|_x \, dP(w|W(u,x)). \end{aligned} \tag{61}$$

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:

$$\begin{aligned} H(u,x) = &\left[\hat{H}(u,x^-,v) + H(v,x)\right] P(W(u,v,x)|W(u,x)) \\ &+ \hat{H}(u,v^-,x)\, P(W(u,v^-,x)|W(u,x)). \end{aligned} \tag{62}$$

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':

$$\begin{aligned}\mathbb{E}[T'] &= \mathbb{E}[S_1 + \cdots + S_N], \\ &= \mathbb{E}[N]\mathbb{E}[S_1], \\ &\leq \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{d(v,x)}.\end{aligned} \tag{66}$$

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:

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

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

$$\begin{aligned}\hat{H}(u, v^-, x) &= \int_{W(u,v^-,x)} |w|_x \, dP(w|W(u, v^-, x)), \\ &= \sum_{k=1}^{\infty} \int_{\{w \in W(u,v^-,x) \wedge |w|_x = k\}} |w|_x \, dP(w|W(u, v^-, x)), \\ &= \sum_{k=1}^{\infty} k \int_{\{w \in W(u,v^-,x) \wedge |w|_x = k\}} dP(w|W(u, v^-, x)), \\ &= \sum_{k=1}^{\infty} k\, P(\{w \in W(u, v^-, x) \wedge |w|_x = k\}|W(u, v^-, x)), \\ &\leq \sum_{k=1}^{\infty} k\, P(\{w : |w|_x = k\}|W(u, v^-, x)), \\ &= \sum_{k=1}^{\infty} k\, \frac{P(\{w : |w|_x = k \wedge w \in W(u, v^-, x)\})}{P(W(u, v^-, x))}, \\ &\leq \sum_{k=1}^{\infty} k(1-\alpha)^k \frac{1}{P(W(u, v^-, x))}, \\ &= \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\frac{1}{P(W(u, v^-, x))}.\end{aligned} \tag{69}$$

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

$$\begin{aligned}\hat{H}(u, v^-, x)\, P(W(u, v^-, x)|W(u,x)) &\leq \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\frac{P(W(u, v^-, x)|W(u,x))}{P(W(u, v^-, x))}, \\ &= \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\frac{1}{P(W(u,x))}.\end{aligned} \tag{70}$$

$P(W(u,x))$ 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 cover time and edge cover time 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 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 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[9] 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. □

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.6.7. 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:

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

We remark that a random walk neural network $X_\theta(\cdot)$ 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

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

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 $\{\mathbf{z}\} \coloneqq \mathrm{Range}(q)$, and consider a decoding function $\psi : \{\mathbf{z}\} \to \mathbb{G}_n$ that takes the record $\mathbf{z} \coloneqq q(v_0 \to \cdots \to v_l, G)$ of a given random walk $v_{[\cdot]}$ and outputs the graph $\psi(\mathbf{z}) \in \mathbb{G}_n$ composed of all recorded vertices $V(H) \coloneqq \{\mathrm{id}(v_t) : v_t \in \{v_0, ..., v_l\}\}$ 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 $\mathrm{id}(\cdot)$:*

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

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

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

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 neighborhoods, and $v_{[\cdot]}$ has visited all vertices of G.*

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 neighborhoods are used to record the induced subgraph $G[V(G)] = G$. □

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. *A random walk neural network $X_\theta(\cdot)$ 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 neighborhoods 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 $\{\mathbf{z}\} \coloneqq \mathrm{Range}(q)$ as follows:

$$\phi' \coloneqq \phi \circ \psi, \tag{87}$$

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

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

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 $\{\mathbf{z}\} \coloneqq \mathrm{Range}(q)$, 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:

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

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 $\mathbf{z}$:

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

and hence:

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

Combining Equations (91) and (94), we have:

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

which can be written as follows:

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

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:

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

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 neighborhoods, except that the edge cover time is changed to the cover time $C_V(G)$ (Equation (16)). This is because neighborhood recording automatically records the induced subgraph of visited vertices, thus visiting all vertices implies recording all edges, $G[V(G)] = G$. □

A.6.8. Proof of Theorem 3.4 (Section 3.1)

Theorem 3.4. *A random walk neural network $X_\theta(\cdot)$ 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 neighborhoods 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 $\mathrm{id}(x)$ whose shortest path distance from the starting vertex $\mathrm{id}(v) = \mathrm{id}(v_0) = 1$ 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.6.9. Proof of Theorem 3.5 (Section 3.2)

Theorem 3.5. *The simple random walk neural network 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.6.10. Proof of Theorem 3.6 (Section 3.2)

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

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

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:

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

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. □

A.7. Limitations and Future Work

Our current main limitation in practice is the cost of performing training and inference with language models, especially on long random walks. We believe efficient fine-tuning with e.g. low-rank adaptation (Hu et al., 2022; Chen et al., 2023a) may help overcome the issue. Also, since the arXiv citation network experimented in this work is homophilic, a future direction would be to test whether our approach works for transductive classification on heterophilic graphs.

On the theoretical side, our universality proof uses long random walks (e.g. $O(n^2)$) such that the whole graph of interest is covered with a high probability. This leaves a question on fine-grained analysis on classes of graphs that can be recognized by our approach efficiently, i.e. using shorter walks. It is known that certain graph information such as the eigen-spectrum can be inferred efficiently from a random walk, for example by observing its returns (Benjamini et al., 2006). Similar idea permeates sublinear graph algorithms for estimating graph statistics (Ben-Hamou et al., 2018). We believe studying these directions would be fruitful in developing a fine-grained complexity theory for random walk neural networks.

A potential challenge of our approach is that the reader neural network has to learn to recognize extremely varying "views" of graphs, as there may exist up to an exponential number of different random walk records that encode the same graph. Interestingly, our empirical results suggest that transformer language models are capable of doing so to some degree, e.g. see examples in Figures 9, 10, and 11. Similar observations were made by Kim et al. (2023), where a vision transformer trained to recognize randomly permuted adjacency matrices of graphs with at most 188 vertices exhibited a non-trivial performance. Studying the capability of large transformers to learn under highly varying views of data would be an interesting direction.

Lastly, a 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.