Non-convolutional Graph Neural Networks

Yuanqing Wang
Center for Data Science
and Simons Center
for Computational Physical Chemistry
New York University
New York, N.Y. 10004
wangyq@wangyq.net

Kyunghyun Cho
Center for Data Science, New York University
and Prescient Design, Genetech
New York, N.Y. 10004
kc119@nyu.edu

Abstract

Rethink convolution-based graph neural networks (GNN)—they characteristically suffer from limited expressiveness, over-smoothing, and over-squashing, and require specialized sparse kernels for efficient computation. Here, we design a simple graph learning module entirely free of convolution operators, coined *random walk with unifying memory* (RUM) neural network, where an RNN merges the topological and semantic graph features along the random walks terminating at each node. Relating the rich literature on RNN behavior and graph topology, we theoretically show and experimentally verify that RUM attenuates the aforementioned symptoms and is more expressive than the Weisfeiler-Lehman (WL) isomorphism test. On a variety of node- and graph-level classification and regression tasks, RUM not only achieves competitive performance, but is also robust, memory-efficient, scalable, and faster than the simplest convolutional GNNs.

1 Introduction: Convolutions in GNNs

Graph neural networks (GNNs) [1, 2, 3, 4, 5]—neural models operating on representations of nodes ($\mathcal{V}$) and edges ($\mathcal{E}$) in a *graph* (denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, with structure represented by the adjacency matrix $A_{ij} = \mathbb{1}[(v_i, v_j) \in \mathcal{E}]$)—have shown promises in a wide array of social and physical modeling applications. Most GNNs follow a *convolutional* scheme, where the D-dimensional node representations $\mathbf{X} \in \mathbb{R}^{|V| \times D}$ are aggregated based on the structure of local neighborhoods:

$$\mathbf{X}' = \hat{A}\mathbf{X}. \tag{1}$$

Here, $\hat{A}$ displays a unique duality—the input features doubles as a compute graph. The difference among *convolutional* GNN architectures, apart from the subsequent treatment of the resulting intermediate representation $\mathbf{X}'$, typically amounts to the choices of transformations ($\hat{A}$) of the original adjacency matrix (A)—the normalized Laplacian for graph convolutional networks (GCN) [1], a learned, sparse stochastic matrix for graph attention networks (GAT) [6], powers of the graph Laplacian for simplifying graph networks (SGN) [7], and the matrix exponential thereof for graph neural diffusion (GRAND) [8], to name a few. For all such transformations, it is easy to verify that permutation equivariance (Equation 9) is trivially satisfied, laying the foundations of data-efficient graph learning. At the same time, this class of methods share common pathologies as well:

Limited expressiveness. (Figure 1) Xu et al. [2] groundbreakingly elucidates that GNNs cannot exceed the expressiveness of Weisfeiler-Lehman (WL) isomorphism test [9]. Worse still, when the support of neighborhood multiset is uncountable, no GNN with a single aggregation function can be

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

	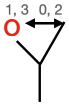	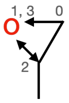	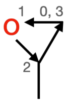	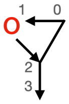	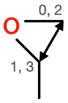	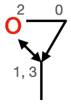	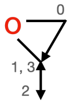	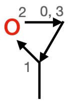
ω_x	COCO	COCO	COCC	COCC	CCCC	CCOC	CCCC	CCOC
ω_u	0101	0121	0120	0123	0101	0121	0121	0120

Table 1: **Schematic illustration of RUM.** All 4-step unbiased random walks from the 2-degree carbon atom in the (hydrogen-omitted) chemical graph of *propylene oxide*, a key precursor for manufacturing polyurethane. The arrows indicate the direction of the walks and numbers the order in which each node is visited. The semantic (ω_x) and topological (ω_u) representations of each walk are shown.

as expressive as the WL test [10]. As such, crucial local properties of graphs meaningful in physical and social modeling, including cycle sizes (Example 8.1) and diameters (Example 8.2) [11], cannot be realized by convolution-based GNNs.

Over-smoothing. (Figure 2) As one repeats the convolution (or Laplacian smoothing) operation (Equation 1), sandwiched by linear and nonlinear transformations, the inter-node dissimilarity, measured by Dirichlet energy,

$$\mathcal{E}(\mathbf{X}) = \frac{1}{N} \sum_{(u,v) \in \mathcal{E}_\mathcal{G}} ||\mathbf{X}_u - \mathbf{X}_\mathbf{v}||^2, \tag{2}$$

will decrease exponentially as the number of message-passing steps l increases [12, 13], $\mathcal{E}(\mathbf{X}^{(l)}) \leq C_1 \exp(-C_2 l)$ with some constants $C_{1,2}$, resulting in node representations only dependent upon the topology, but not the initial embedding.

Over-squashing. (Figure 5) As the number of Laplacian smoothing grows, the receptive field of GNNs increases exponentially, while the dimension of node representation, and thereby the possible combinations of neighborhood environment, stays unchanged [14]. Quantitatively, Topping et al. [15] quantifies this insight using the inter-node Jacobian and relates it to the powers of the adjacency matrix through *sensitivity analysis*:

$$|\frac{\partial \mathbf{X}_v^{(l+1)}}{\partial \mathbf{X}_u^{(0)}}| \leq |\nabla \phi|^{(l+1)} (\hat{A}^{l+1})_{uv}, \tag{3}$$

where ϕ is the node-wise update function, whose Jacobian is typically diminishing. If this Equation 3 converges to zero, the node representation is agnostic to the changes happening l edges away, making convolutional GNNs difficult to learn long-range dependencies.

Main contributions: Non-convolutional GNNs as a joint remedy. In this paper, we propose a variant of GNN that does not engage the convolution operator (Equation 1) at all.[1] Specifically, we stochastically sample a random walk terminating at each node and use the *anonymous experiment* associated with the random walk to describe its topological environment. This is combined with the semantic representations along the walk and fed into a recurrent neural network layer [16] to form the node embedding, which we call the *unifying memory*. We theoretically (§ 4) and experimentally (§ 5) show that the resulting model, termed *random walk with unifying memory* (RUM) relieves the aforementioned symptoms and offers a compelling alternative to the popular convolution-based GNNs.

2 Related works: ways to walk on a graph

Walk-based GNNs. RAW-GNN ([17], compared and outperformed in Table 9) also proposes walk-based representations for representing node neighborhoods, which resembles our model without the *anonymous experiment* (ω_u in Equation 5). CRaWl ([18, 19], outperformed in Table 3) also incorporates a similar structural encoding for random walk-generated subgraphs to feed into an

[1] Code at: `https://github.com/yuanqing-wang/rum/tree/main`

iterative 1-dimensional convolutional network. AWE ([20], Table 3) and Wang et al. [21], like ours, use anonymous experiments for graph-level unsupervised learning and temporal graph learning, respectively. More elaborately, [22] and AgentNet ([23], Table 3) use agent-based learning on random walks and paths on graphs.

Random walk kernel GNNs. In a closely related line of research, RWK ([24], Table 3) employs the reproducing kernel Hilbert space representations in a neural network for graph modeling and GSN ([25], Table 3) counts the explicitly enumerated subgraphs to represent graphs. The subgraph counting techniques intrinsically require prior knowledge about the input graph of a predefined set of node and edge sets. For these works, superior expressiveness has been routinely argued, though usually limited to a few special cases where the WL test fails whereas they do not, and often within the *unlabelled graphs* only.

More importantly, focusing mostly on expressiveness, no aforementioned **walk-based** or **random walk kernel**-based GNNs address the issue of over-smoothing and over-squashing in GNNs. Some of these works are also *de facto* convolutional, as the random walks are only incorporated as features in the message-passing scheme. Interestingly, most of these works are either not applicable to, or have not been tested on, node-level tasks. In the experimental § 5, we show that RUM not only outperforms these baselines on most graph-level tasks (Table 3) but also competes with a wide array of state-of-the-art convolutional GNNs on node-level tests (Table 2). Moreover, random walk kernel-based architectures, which explicitly enumerate random walk-generated subgraphs are typically much slower than WL-equivalent convolutional GNNs, whereas RUM is faster than even the simplest variation of convolutional GNN (Figure 4).

Graph transformers. Graph transformers [26, 27]—neural models that perform attention among all pairs of nodes and encode graph structure via positional encoding—are well-known solutions that are not locally convolutional. Their inductive biases determine that over-smoothing and over-squashing among local neighborhoods are, like RUM, also not prominent.

Because of its all-to-all nature, the runtime complexity of graph transformers, just like that of almost all transformers, contains a quadratic term, w.r.t. the size of the system (number of nodes, edges, or subgraphs). This makes it prohibitively expensive and memory intensive on large social graphs (such as that used in Table 8 with millions of nodes). On smaller social graphs, we show in Tables 9, 4 that graph transformers are experimentally outperformed by RUM.

State-of-the-art methods to alleviate oversmoothing. *Stochastic regularization.* DropEdge ([28], Figure 2) regularizes the smoothing of the node representations by randomly disconnecting edges. Its associated Dirichlet energy indeed decreases slower, though eventually still diminishes as the number of layers increases. *Graph rewiring.* [29, 30] and GPR-GNN ([31], Appendix Table 9) rewire the graph using personalized page rank algorithm [32] and generalized page rank on graphs, respectively. Similar to JKNet [33], they mitigate over-smoothing by allowing direct message passing between faraway nodes. *Constant-energy methods.* Zhao and Akoglu [34], Rusch et al. [35] constrain the pair-wise distance or Dirichlet energy among graphs to be constant. Nevertheless, the non-decreasing energy does not necessarily translate to better performance, as they sometimes come with the sacrifice of expressiveness, as argued in Rusch et al. [13]. *Residual GNNs.* Residual connection [36] can naturally be added to the GNN architecture, such as GCNII ([37], Table 2), to restraint activation to be similar to the input to allow deep networks. They however can make the model less robust to perturbations in the input. In sum, these works have similar, if not compromised expressiveness compared to a barebone GCN.

3 Architecture: combining topologic and semantic trajectories of walks

Random walks on graphs. An unbiased random walk w on a graph $\mathcal{G}$ is a sequence of nodes $w = (v_0, v_1, \ldots)$ with landing probability:

$$P(v_j | (v_0, \ldots, v_{i-1})) = \mathbb{1}[(v_i, v_j) \in \mathcal{E}_\mathcal{G}] / D(v_i), \tag{4}$$

where $D(v_i) = \sum A_{ij}$ is the degree of the node v_i. Walks *originating* from or *terminating* at any given node v can thus be easily generated using this Markov chain. We record the trajectory of embeddings associated with the walk as $\omega_x(w) = (\mathbf{X}_i) = (\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_l)$. In this paper, we only consider finite-long l-step random walk $|w| = l \in \mathbb{Z}^+$. In our implementation, the random walks are sampled *ad hoc* during each training and inference step directly on GPU using Deep Graph Library [38] (see Appendix § A). Moreover, the *walk* considered here is not necessarily a *path*, as repeated traversal of the same node $v_i = v_j, i \neq j$ is not only permitted, but also crucial to effective topological representation, as discussed below.

Anonymous experiment. We use a function describing the topological environment of a walk, termed *anonymous experiment* [39], $\omega_u(w) : \mathbb{R}^l \rightarrow \mathbb{R}^l$ that records **the first unique occurrence of a node in a walk** (Appendix Algorithm C). To put it plainly, we label a node as the number of *unique* nodes insofar traversed in a walk if the node has not been traversed, and reuse the label otherwise. Practically, this can be implemented using any tensor-accelerating framework *in one line* (`w` is the node sequence of a walk) and trivially parallelized [2]: `(1*(w[..., :,None]==w[..., None,:])).argmax(-1)`

Unifying memory: combining semantic and topological representations. Given any walk w, we now have two sequences $\omega_x(w)$ and $\omega_u(w)$ describing the *semantic* and *topological* (as we shall show in the following sections) features of the walk. We project such sequential representations onto a latent dimension to combine them (illustrated in Table 1):

$$h(w) = f(\phi_x(\omega_x(w)), \phi_u(\omega_u(w))), \tag{5}$$

where $\phi_x : \mathbb{R}^{l \times D} \rightarrow \mathbb{R}^{D_x}$ maps the sequence of semantic embeddings generated by a l-step walk to a fixed D_x-dimensional latent space, $\phi_u : \mathbb{R}^l \rightarrow \mathbb{R}^{D_u}$ maps the indicies sequence to another latent space D_u, and $f : \mathbb{R}^{D_x} \oplus \mathbb{R}^{D_u} \rightarrow \mathbb{R}^D$ combines them. We call Equation 5 the *unifying memory* of a random walk. Subsequently, the node representations can also be formed as the average representations of l-step (l being a hyperparameter) walks *terminating* (for the sake of gradient aggregation) at that node:

$$\psi(v) = \sum_{\{w\}, |w|=l, w_l=v} p(w)h(w), \tag{6}$$

which can be stochastically sampled with unbiased Monte Carlo gradient and used in downstream tasks as such node classification and regression. We note that this is the only time we perform SUM or MEAN operations. Unlike other GNNs incorporating random walk-generated features (which are sometimes still convolutional and iterative), we do not iteratively pool representations within local neighborhoods. The likelihood of the data written as:

$$P(y|\mathcal{G}, \mathbf{X}) = \sum_{\{w\}, |w|=l, w_l=v} p(w)p(y|\mathcal{G}, \mathbf{X}, w) \tag{7}$$

The node representation can be summed

$$\Psi(\mathcal{G}) = \sum_{v \in \mathcal{V} \subseteq \mathcal{G}} \psi(v) \tag{8}$$

to form global representations for graph classification and regression. We call ψ in Equation 6 and Ψ in Equation 8 the node and graph output representations of RUM.

Layer choices. Obvious choices to model f include a feed-forward neural network after concatenation, and ϕ_x, ϕ_u recurrent neural networks (RNNs). This implies that, different from most convolutional GNNs, parameter sharing is natural and the number of parameters is going to stay constant as the model incorporates a larger neighborhood. Compared to dot product-based, transformer-like modules [40], RNNs not only have linear complexity (see detailed discussions below) w.r.t. the sequence length but also naturally encodes the inductive bias that nodes closer to the origin have stronger impact on the representation. The gated recurrent unit (GRU)[16] variant is used everywhere in this paper. Additional regularizations are described in Appendix § B.1.

[2] Note that this particular implementation introduces an intermediate $\mathcal{O}(l^2)$ complexity term, though it is empirically faster than the linear-complexity naive implementation, since only integer indices are involved and thus the footprint is negligible.

Runtime complexity. To generate random walks for one node has the runtime complexity of $\mathcal{O}(1)$, and for a graph $\mathcal{O}(|\mathcal{V}|)$, where $|\mathcal{V}|$ is the number of nodes in a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. To calculate the *anonymous experiment*, as shown in Appendix Algorithm C, has $\mathcal{O}(1)$ complexity (also see Footnote 2). If we use linear-complexity models, such as RNNs, to model ϕ_x, ϕ_u, the overall complexity is $\mathcal{O}(|\mathcal{V}|lkD)$ where l is the length of the random walk, k the samples used to estimate Equation 6, and D the latent size of the model (assumed uniform). Note that different from convolutional GNNs, RUM does not depend on the number of edges $|\mathcal{E}|$ (which is usually much larger than $|\mathcal{V}|$) for runtime complexity, and is therefore agnostic to the *sparsity* of the graph. See Figure 4 for an empirical verification of the time complexity. In Appendix Table 8, we show, on a large graph, the overhead introduced by generating random walks and computing anonymous experiments accounts for roughly $1/1500$ of the memory footprint and $1/8$ of the wall time.

Mini-batches. RUM is naturally compatible with mini-batching. For convolutional GNNs, large graphs that do not fit into the GPU memory have traditionally been a challenge, as all neighbors are required to be present and boundary conditions are hard to define [41]. RUM, on the other hand, can be inherently applied on subsets of nodes of a large graph without any alteration in the algorithm—the random walks can be generated on a per-node basis, and the FOR loop in Algorithm C can be executed sequentially, in parallel, or on subsets. Empirically, in Appendix Table 8, RUM can be naturally scaled to huge graphs.

4 Theory: RUM as a joint remedy.

We have insofar designed a new graph learning framework—convolution-free graph neural networks (GNNs) that cleanly represent the semantic (ω_x) and topological (ω_u) features of graph-structured data before unifying them. First, we state that RUM is permutation equivariant,

Remark 1 (Permutation equivariance). For any permutation matrix P, we have

$$P\mathbf{X}_v(\mathcal{G}) = \mathbf{X}_v(P(\mathcal{G})), \tag{9}$$

which sets the foundation for the data-efficient modeling of graphs. Next, we theoretically demonstrate that this formulation jointly remedies the common pathologies of the convolution-based GNNs by showing that: (a) the *topological* representation ω_u is more expressive than convolutional-GNNs in distinguishing non-isomorphic graphs; (b) the *semantic* representation ω_x no longer suffers from over-smoothing and over-squashing.

4.1 RUM is more expressive than convolutional GNNs.

For the sake of theoretical arguments in this section, we assume that in Equation 5:

Assumption 2. ϕ_x, ϕ_u, f are universal and injective.

Assumption 3. Graph $\mathcal{G}$ discussed in this section is always connected, unweighted, and undirected.

Assumption 2 is easy to satisfy for feed-forward neural networks [42] and RNNs [43]. Note that RUM can be easily extended to weighted graphs by sampling a biased random walk with edge weights w_{ij} and keeping rest of the algorithm the same: $P(v_j|(v_0, ..., v_{i-1})) \propto I[(v_i, v_j) \in E_G] * w_{ij}/D(v_i)$. Composing injective functions, we remark that $h(w)$ is also injective w.r.t. $\omega_x(w)$ and $\omega_u(w)$; despite of Assumption 3, our analysis can be extended to disjointed graphs by restricting the analysis to the connected regions in a graph. Under such assumptions, we show, in **Remark** 8 (deferred to the Appendix), that ψ is **injective**, meaning that nodes with different random walks will have different distributions of representations $\psi(v_1) \neq \psi(v_2)$. We also refer the readers to the **Theorem 1** in Micali and Zhu [39] for a discussion on the representation power of *anonymous experiments* on *unlabelled* graphs. Combining with the semantic representations and promoting the argument from a node level to a graph level, we arrive at:

Theorem 4 (RUM can distinguish non-isomorphic graphs)**.** *Up to the Reconstruction Conjecture [44], RUM with sufficiently long l-step random walks can distinguish non-isomorphic graphs satisfying Assumption 3.*

The main idea of the proof of Theorem 4 (in Appendix § D.2) involves explicitly enumerating all possible non-isomorphic structures for graphs with 3 nodes and showing, by induction, that if

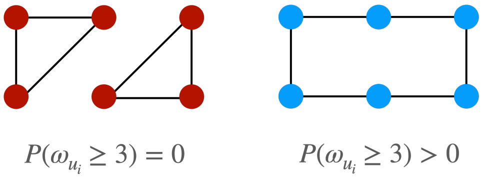

Figure 1: RUM can (in closed form), whereas the Weisfeiler-Lehman (WL) isomorphism test and WL-equivalent GNNs *cannot*, distinguish these two graphs—**an illustration of Example 8.1**.

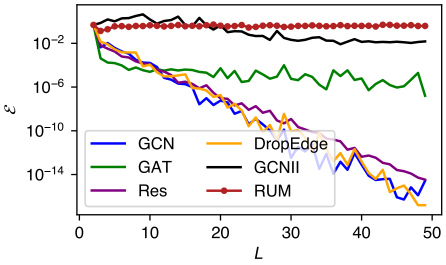

Figure 2: **RUM alleviates over-smoothing**. Dirichlet energy ($\mathcal{E}$) on Cora [47] graph plotted against L, the number of steps or layers.

the theorem stands for graph of $N - 1$ size it also holds for N-sized graphs. We also show in Appendix § D.1 that a number of key graph properties such as cycle size (Example 8.1) and radius (Example 8.2) that convolutional GNNs struggle [45, 46] to learn can be analytically expressed using ω_u. As these are solely functions of ω_x, they can be approximated arbitrarily well by universal approximators. These examples are special cases of the finding that RUM is stricly more expressive than Weisfeiler-Lehman isomorphism test [9]:

Corollary 4.1 (RUM is more expressive than WL-test)**.** *Up to the Reconstruction Conjecture, two graphs with G_1, G_2 labeled as non-isomorphic by the k-dimensional Weisfeiler-Lehman (k-WL) isomorphism test, is the necessary, but not sufficient condition that the representations resulting from RUM with walk length k are also different.*

$$\Psi(\mathcal{G}_1) \neq \Psi(\mathcal{G}_2) \tag{10}$$

Thus, due to Xu et al. [2], RUM is also more expressive than convolutional GNNs in distinguishing non-isomorphic graphs. This also confirms the intuition that RUM with longer walks are more expressive. (See Figure 3 on an empirical evaluation.) The proof of this corollary is straightforward to sketch—even if we only employ the embedding trajectory ω_x, it would have the effect of performing the function ϕ_x in Equation 5 on each traversal of the WL expanded trees.

4.2 RUM alleviates over-smoothing and over-squashing

Over-smoothing refers to the phenomenon where the node dissimilarity (e.g., measured by Dirichlet energy in Equation 2) decreases exponentially and approaches zero with the repeated rounds of message passing. Cai and Wang [12] relates Dirichlet energy directly with the convolutional operator:

Lemma 3.1 from Cai and Wang [12].

$$\mathcal{E}((1 - \tilde{\Delta})\mathbf{X}) \leq (1 - \lambda)^2 \mathcal{E}(\mathbf{X}) \tag{11}$$

where λ is the smallest non-zero eigenvalue of $\tilde{\Delta}$, the normalized Laplacian of a graph.

Free of convolution operators, it seems only natural that RUM does not suffer from this symptom (Figure 2). We now formalize this intuition by first restricting ourselves to a class of *non-contractive* mappings for f in Definition 5.

Definition 5. A map f is non-contractive on region Ω if $\exists \alpha \in [1, +\infty)$ such that $|f(x) - f(y)| \geq \alpha ||x - y||, \forall x, y \in \Omega$.

A line of fruitful research has been focusing on designing non-contractive RNNs [48, 49], and to be non-contractive is intimately linked with desirable properties such as preserving the long-range information content and non-vanishing gradients. From this definition, it is easy to show that, for each sampled random walk in Equation 5, the Dirichlet energy is greater than its input. One only needs to verify that the integration in Equation 6 does not change this to arrive at:

Lemma 6 (RUM alleviates over-smoothing.)**.** *If ϕ_x, f are non-contractive w.r.t. all elements in the sequence, the expected Dirichlet energy of the corresponding RUM node representation in Equation 6 is greater than its initial value*

$$\mathrm{E}(\mathcal{E}(\psi(\mathbf{X}))) \geq \mathcal{E}(\mathbf{X}). \tag{12}$$

This implies that the expectation of Dirichlet energy does not diminish even when $l \rightarrow +\infty$, as it is bounded by the Dirichlet energy of the initial node representation, which is consistent with the trend shown in Figure 2, although the GRU is used out-of-box without constraining it to be explicitly non-contractive.

RUM alleviates over-squashing is deferred to Appenxix § B.2, where we verify that the inter-node Jacobian $|\partial \mathbf{X}_v^{(l+1)} / \partial \mathbf{X}_u^{(0)}|$ decays slower as the distance between u, v grows vis-à-vis the convolutional counterparts. Briefly, although RUM does not address the information bottleneck with exponentially growing receptive field (the $1/(\hat{A}^{l+1})_{uv}$ term in Equation 16), it nevertheless can have a non-vanishing (nor exploding) gradient from the aggregation function ($|\nabla \phi_x|$).

5 Experiments

On a wide array of real-world node- and graph-level tasks, we benchmark the performance of RUM to show its utility in social and physical modeling. Next, to thoroughly examine the performance of RUM, we challenge it with carefully designed illustrative experiments. Specifically, we ask the following questions in this section, with **Q1**, **Q2**, and **Q3** already theoretically answered in § 4: **Q1:** Is RUM more expressive than convolutional GNNs? **Q2:** Does RUM alleviate over-smoothing? **Q3:** Does RUM alleviate over-squashing? **Q4:** Is RUM slower with convolutional GNNs? **Q5:** Is RUM robust? **Q6:** How does RUM scale up to huge graphs? **Q7:** What components of RUM are contributing most to the performance of RUM?

Real-world benchmark performance. For node classification, we benchmark our model on the popular Planetoid citation datasets [47], as well as the coauthor [50] and co-purchase [51] datasets common in social modeling. Additionally, we hypothesize that RUM, without the smoothing operator, will perform competitively on heterophilic datasets [52]—we test this hypothesis. For graph classification, we benchmark on the popular TU dataset [53]. We also test the graph regression performance on molecular datasets in MoleculeNet [54] and Open Graph Benchmark [55]. In sum, RUM almost always outperforms, is within the standard deviation of, the state-of-the-art architectures, as shown in Tables 2, 3, 4, 5, as well as in Tables 6, 7, 8, 9 moved to the Appendix due to space constraint.

On sparsity: the subpar performance on the Computer dataset. The most noticeable exception to the good performance of RUM is that on the Computer co-purchase [51] dataset, where RUM is outperformed even by GCN and GAT. This dataset is very dense with an average node degree ($|\mathcal{E}|/|\mathcal{V}|$) of 18.36, the highest among all datasets used in this paper. As the variance of the node embedding (Equation 6) scales with the average degree, we hypothesize that dense graphs with very high average node degrees would have high-variance representations from RUM.

On the other hand, RUM outperforms *all* models surveyed in two large-scale benchmark studies on molecular learning, GAUCHE [56] and MoleculeNet [54]. The atoms in the molecules always have a degree of $2 \sim 4$ with intricate subgraph structures like small rings. This suggests the utility of *unifying memory* in chemical graphs and furthermore chemical and physical modeling.

Graph isomorphism testing (Q1). Having illustrated in § 4 that RUM can distinguish non-isomorphic graphs, we experimentally test this insight on the popular Circular Skip Link dataset [60, 61]. Containing 4-regular graph with edges connected to form a cycle and containing skip-links between nodes, this dataset is artificially synthesized in Murphy et al. [60] to create an especially challenging task for GNNs. As shown in Appendix Table 7, all convolutional GNNs fail to perform better than a constant baseline (there are 10 classes uniformly distributed). 3WLGNN [62], a higher-order GNN of at least $\mathcal{O}(2)$ complexity that operates on explicitly enumerated triplets of graphs, can distinguish these highly similar 4-regular graphs by comparing subgraphs. RUM, with $\mathcal{O}(|\mathcal{N}|)$ linear complexity, achieves similarly high accuracy. One can think of RUM as a stochastic approximation

	Cora	CiteSeer	PubMed	Coathor CS	Computer	Photo
GCN[1]	81.5	70.3	79.0	91.1$_{\pm 0.5}$	82.6$_{\pm 2.4}$	91.2$_{\pm 1.2}$
GAT[6]	83.0$_{\pm 0.7}$	72.5$_{\pm 0.7}$	79.0$_{\pm 0.3}$	90.5$_{\pm 0.6}$	78.0$_{\pm 19.0}$	85.7$_{\pm 20.3}$
GraphSAGE[4]	77.4$_{\pm 1.0}$	67.0$_{\pm 1.0}$	76.6$_{\pm 0.8}$	85.0$_{\pm 1.1}$		90.4$_{\pm 1.3}$
MoNet[57]	81.7$_{\pm 0.5}$	70.0$_{\pm 0.6}$	78.8$_{\pm 0.4}$	90.8$_{\pm 0.6}$	83.5$_{\pm 2.2}$	91.2$_{\pm 2.3}$
GCNII[37]	85.5$_{\pm 0.5}$	73.4$_{\pm 0.6}$	80.3$_{\pm 0.4}$			
PairNorm[34]	81.1	70.6	78.2			
GraphCON[35]	84.2$_{\pm 1.3}$	74.2$_{\pm 1.7}$	79.4$_{\pm 1.3}$			
RUM	84.1$_{\pm 0.9}$	75.5$_{\pm 0.5}$	82.2$_{\pm 0.2}$	93.2$_{\pm 0.0}$	77.8$_{\pm 2.3}$	92.7$_{\pm 0.1}$

Table 2: **Node classification** test set accuracy ↑ and standard deviation.

	IMDB-B	MUTAG	PROTEINS	PTC	NCI1
RWK[24]		79.2$_{\pm 2.1}$	59.6$_{\pm 0.1}$	55.9$_{\pm 0.3}$	
GK[58]		81.4$_{\pm 1.7}$	71.4$_{\pm 0.3}$	55.7$_{\pm 0.5}$	62.5$_{\pm 0.3}$
WLK[59]	73.8$_{\pm 3.9}$	90.4$_{\pm 5.7}$	75.0$_{\pm 3.1}$	59.9$_{\pm 4.3}$	86.0$_{\pm 1.8}$
AWE[20]	74.5$_{\pm 5.9}$	87.8$_{\pm 9.8}$			
GIN[2]	75.1$_{\pm 5.1}$	90.0$_{\pm 8.8}$	76.2$_{\pm 2.6}$	66.6$_{\pm 6.9}$	82.7$_{\pm 1.6}$
GSN[25]	77.8$_{\pm 3.3}$	92.2$_{\pm 7.5}$	76.6$_{\pm 5.0}$	68.2$_{\pm 7.2}$	83.5$_{\pm 2.0}$
CRaWl[19]	73.4$_{\pm 2.1}$	90.4$_{\pm 7.1}$	76.2$_{\pm 3.7}$	68.0$_{\pm 6.5}$	
AgentNet[23]	75.2$_{\pm 4.6}$	93.6$_{\pm 8.6}$	76.7$_{\pm 3.2}$	67.4$_{\pm 5.9}$	
RUM	81.1$_{\pm 4.5}$	91.0$_{\pm 7.1}$	77.3$_{\pm 3.8}$	69.8$_{\pm 6.3}$	81.7$_{\pm 1.4}$

Table 3: **Binary graph classification** test set accuracy ↑.

of the higher-order GNN, with all of its explicitly enumerated subgraphs being identified by RUM with a probability that decreases with the complexity of the subgraph.

Effects of walk lengths and number of samples on performance (Q2, Q3). Having studied the relationship between inference speed and the walk lengths and number of samples, we furthermore study its impact on performance. Using Cora [47] citation graph and vary the walk lengths and number of samples from 1 to 9, where the performance of RUM improves as more samples are taken and longer walks are employed, though more than 4 samples and walks longer than $L > 4$ yield qualitatively satisfactory results; this empirical finding has guided our hyperparameter design. In Figure 2, we also compare the Dirichlet energy of RUM-generated layer representations with not only baselines GCN [1] and GAT [6], but also strategies to alleviate over-smoothing discussed in § 2, namely residual connection and stochastic regularization [37, 28, 63], and show that when L gets large, only RUM can maintain Dirichlet energy. Traditionally, since Kipf and Welling [1] (see its Figure 5 compared to Figure 3), the best performance on Cora graph was found with 2 or 3 message-passing rounds, since mostly local interactions are dominating the classification, and more rounds of message-passing almost always lead to worse performance. As theoretically demonstrated in § 4.2, RUM is not as affected by these symptoms. Thus, RUM is especially appropriate for modeling long-range interactions in graphs without sacrificing local representation power.

	Texas	Wisc.	Cornell
GCN[1]	55.1$_{\pm 4.2}$	51.8$_{\pm 3.3}$	60.5$_{\pm 4.8}$
GAT[6]	52.2$_{\pm 6.6}$	51.8$_{\pm 3.1}$	60.5$_{\pm 5.3}$
GCNII[37]	77.6$_{\pm 3.8}$	80.4$_{\pm 3.4}$	77.9$_{\pm 3.8}$
Geom-GCN[52]	66.8$_{\pm 2.7}$	64.5$_{\pm 3.7}$	60.5$_{\pm 3.7}$
PairNorm[34]	60.3$_{\pm 4.3}$	48.4$_{\pm 6.1}$	58.9$_{\pm 3.2}$
GPS[26]	75.4$_{\pm 1.5}$	78.0$_{\pm 2.9}$	65.4$_{\pm 5.7}$
Graphomer [27]	76.8$_{\pm 1.8}$	77.7$_{\pm 2.0}$	68.4$_{\pm 1.7}$
RUM	80.0$_{\pm 7.0}$	85.8$_{\pm 4.1}$	71.1$_{\pm 5.6}$

Table 4: **Node classification** test set accuracy ↑ and standard deviation on heterophilic [52] datasets.

	ESOL	FreeSolv	Lipophilicity
GAUCHE[56]	0.67$_{\pm 0.01}$	0.96$_{\pm 0.01}$	0.73$_{\pm 0.02}$
MoleculeNet [54]	0.58	1.15	0.80
RUM	0.62$_{\pm 0.06}$	0.96$_{\pm 0.24}$	0.66$_{\pm 0.01}$

Table 5: **Graph regression** RMSE ↓ compared with the *best* model studied in two large-scale benchmark studies on OGB [55] and MoleculeNet [54] datasets.

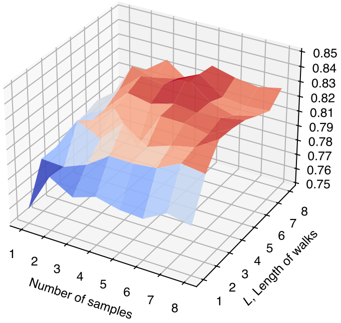

Figure 3: **Impact of number of samples and walk length.** Test classification accuracy of Cora [47] with varying numbers of samples and walk length.

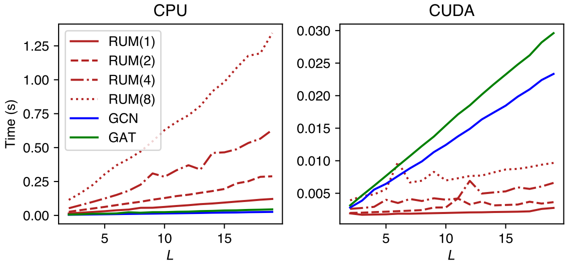

Figure 4: **RUM is faster than convolutional GNNs on GPU.** Inference time over the Cora [47] graph on CPU and CUDA devices, respectively, plotted against L, the number of message-passing steps or equivalently the length of random walks. Numbers in the bracket indicate the number of sampled random walks drawn.

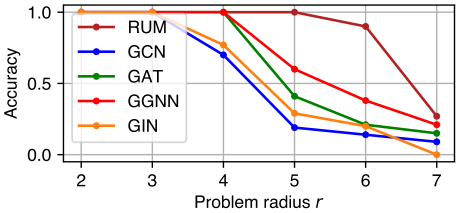

Figure 5: **Long-range neighborhood matching** training accuracy ↑ [14] with 32 unit models.

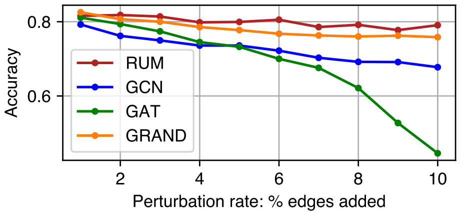

Figure 6: **Robustness analysis**. Accuracy ↑ on Cora [47] dataset with % fictitious edges added to the graph.

Long-range neighborhood matching (Q3). To verify that RUM indeed alleviates over-squashing (§ B.2), in Figure 5, we adopt the tree neighborhood matching synthetic task in Alon and Yahav [14] where binary tree graphs are proposed with the label *and the attributes* of the root matching a faraway leave. The full discussion is moved to the Appendix § B.3.

Speed (Q4). Though both have linear runtime complexity (See § 3), intuitively, it might seem that RUM would be slower than convolution-based GCN due to the requirement of multiple random walk samples. This is indeed true for CPU implementations shown in Figure 4 left. When equipped with GPUs (specs in Appendix § A), however, RUM is significantly faster than even the simplest convolutional GNN—GCN [1]. It is worth mentioning that the GCN and GAT [6] results were harvested using the heavily optimized Deep Graph Library [38] sparse implementation whereas RUM is implemented naïvely in PyTorch [64], though the popular GRU component [16] have already undergone CUDA-specific optimization.

Robustness to attacks (Q5). With the stochasticity afforded by the random walks, it is natural to suspect RUM to be robust. We adopt the robustness test from Feng et al. [65] and attack by randomly adding fake edges to the Cora [47] graph and record the test set performance in Figure 6. Indeed, RUM is much more robust than traditional convolutional GNNs including GCN [1] and GAT [6] and is even slightly more robust than the convolutional GNN specially designed for robustness [65], with the performance only decreased less than 10% with 10% fake edges added.

Scaling to huge graphs (Q6) and Ablation study (Q7) are deferred to Appendix § B.5.

6 Conclusions

We design an innovative GNN that uses an RNN to unify the semantic and topological representations along stochastically sampled random walks, termed *random walk with unifying memory* (RUM) neural networks. Free of the convolutional operators, our methodology does not suffer from symptoms characteristic of Laplacian smoothing, including limited expressiveness, over-smoothing, and over-squashing. Most notably, our method is more expressive than the Weisfeiler-Lehman isomorphism test and can distinguish all non-isomorphic graphs up to the *reconstruction conjecture.* Thus, it is more expressive than all of convolutional GNNs equivalent to the WL test, as we demonstrate theoretically in § 4. RUM is significantly faster on GPUs than even the simplest convolutional GNNs (§ 5) and shows superior performance across a plethora of node- and graph-level benchmarks.

Limitations. *Very dense graphs.* As evidenced by the underwhelming performance of the Computer [51] dataset and discussed in § 5, RUM might suffer from high variance with very dense graphs (average degree over 15). *Tottering.* In our implementation, we have not ruled out the 2-cycles from the random walks, as that would require specialized implementation for walk generation. This, however, would reduce the average information content in a fixed-length random walk (known as tottering [66]). We plan to characterize the effect of excluding these walks. *Biased walk.* Here, we have only considered unbiased random walk, whereas biased random walk might display more intriguing properties as they tend to explore faraway neighborhoods more effectively [17]. *Directed graphs.* Since we have only developed RUM for undirected graph (random walk up to a random length is not guaranteed to exist for directed graphs), we would have to symmetrize the graph to perform on directed graphs (such as the heterophilic datasets [52]); this create additional information loss and complexity.

Future directions. *Theoretical.* We plan to expand our theoretical framework to account for the change in layer width and depth to derive analytical estimates for realizing key graph properties. *Applications.* Random walks are intrinsically applicable to uncertainty-aware learning. We plan to incorporate the uncertainty naturally afforded by the model to design active learning models. On the other hand, the physics-based graph modeling field is also heavily dominated by convolutional GNNs. Inspired by the superior performance of RUM on chemical datasets, we plan to apply our method in drug discovery settings [67, 68, 69, 70, 71] and furthermore on the equivariant modeling of n-body physical systems [72].

Impact statement. We here present a powerful, robust, and efficient learning algorithm on graphs. Used appropriately, this algorithm might advance the modeling of social [73] and physical [74] systems, which can oftentimes modeled as graphs. As with all graph machine learning methods, negative implications may be possible if used in the design of explosives, toxins, chemical weapons, and overly addictive recreational narcotics.

References

[1] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

[2] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[3] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[4] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[7] Felix Wu, Tianyi Zhang, Amauri H. Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. *CoRR*, abs/1902.07153, 2019. URL `http://arxiv.org/abs/1902.07153`.

[8] Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Stefan Webb, Emanuele Rossi, and Michael M. Bronstein. GRAND: graph neural diffusion. *CoRR*, abs/2106.10934, 2021. URL `https://arxiv.org/abs/2106.10934`.

[9] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein.

[10] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets, 2020.

[11] Vikas K. Garg, Stefanie Jegelka, and Tommi S. Jaakkola. Generalization and representational limits of graph neural networks. *CoRR*, abs/2002.06157, 2020. URL `https://arxiv.org/abs/2002.06157`.

[12] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks, 2020.

[13] T. Konstantin Rusch, Michael M. Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks, 2023.

[14] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *CoRR*, abs/2006.05205, 2020. URL `https://arxiv.org/abs/2006.05205`.

[15] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature, 2022.

[16] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

[17] Di Jin, Rui Wang, Meng Ge, Dongxiao He, Xiang Li, Wei Lin, and Weixiong Zhang. Raw-gnn: Random walk aggregation based graph neural network, 2022.

[18] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Graph learning with 1d convolutions on random walks. *CoRR*, abs/2102.08786, 2021. URL `https://arxiv.org/abs/2102.08786`.

[19] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing, 2023.

[20] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings, 2018.

[21] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks, 2022.

[22] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning, 2018.

[23] Karolis Martinkus, Pál András Papp, Benedikt Schesch, and Roger Wattenhofer. Agent-based graph neural networks, 2023.

[24] Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16211–16222. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/ba95d78a7c942571185308775a97a3a0-Paper.pdf`.

[25] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, January 2023. ISSN 1939-3539. doi: 10.1109/tpami.2022.3154319. URL `http://dx.doi.org/10.1109/TPAMI.2022.3154319`.

[26] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer, 2023. URL `https://arxiv.org/abs/2205.12454`.

[27] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *CoRR*, abs/2106.05234, 2021. URL `https://arxiv.org/abs/2106.05234`.

[28] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. The truly deep graph convolutional networks for node classification. *CoRR*, abs/1907.10903, 2019. URL `http://arxiv.org/abs/1907.10903`.

[29] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank, 2022.

[30] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning, 2022.

[31] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Joint adaptive feature smoothing and topology extraction via generalized pagerank gnns. *CoRR*, abs/2006.07988, 2020. URL `https://arxiv.org/abs/2006.07988`.

[32] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking : Bringing order to the web. In *The Web Conference*, 1999. URL `https://api.semanticscholar.org/CorpusID:1508503`.

[33] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *CoRR*, abs/1806.03536, 2018. URL `http://arxiv.org/abs/1806.03536`.

[34] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *CoRR*, abs/1909.12223, 2019. URL `http://arxiv.org/abs/1909.12223`.

[35] T. Konstantin Rusch, Benjamin Paul Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael M. Bronstein. Graph-coupled oscillator networks. *CoRR*, abs/2202.02296, 2022. URL `https://arxiv.org/abs/2202.02296`.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[37] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks, 2020.

[38] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2020.

[39] Silvio Micali and Zeyuan Allen Zhu. Reconstructing markov processes from independent and anonymous experiments. *Discrete Applied Mathematics*, 200:108–122, 2016. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2015.06.035. URL `https://www.sciencedirect.com/science/article/pii/S0166218X15003212`.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[41] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. A comprehensive study on large-scale graph training: Benchmarking and rethinking, 2023.

[42] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[43] Anton Maximilian Schäfer and Hans Georg Zimmermann. Recurrent neural networks are universal approximators. In *Artificial Neural Networks–ICANN 2006: 16th International Conference, Athens, Greece, September 10-14, 2006. Proceedings, Part I 16*, pages 632–640. Springer, 2006.

[44] Paul J Kelly. A congruence theorem for trees. 1957.

[45] Andreas Loukas. What graph neural networks cannot learn: depth vs width. *CoRR*, abs/1907.03199, 2019. URL `http://arxiv.org/abs/1907.03199`.

[46] Vikas K. Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks, 2020.

[47] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *CoRR*, abs/1603.08861, 2016. URL http://arxiv.org/abs/1603.08861.

[48] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[49] António H. Ribeiro, Koen Tiels, Luis A. Aguirre, and Thomas Schön. Beyond exploding and vanishing gradients: analysing rnn training using attractors and smoothness. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2370–2380. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/ribeiro20a.html.

[50] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[51] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[52] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks, 2020.

[53] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs, 2020.

[54] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.*, 9:513–530, 2018. doi: 10.1039/C7SC02664A. URL http://dx.doi.org/10.1039/C7SC02664A.

[55] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *CoRR*, abs/2005.00687, 2020. URL https://arxiv.org/abs/2005.00687.

[56] Ryan-Rhys Griffiths, Leo Klarner, Henry B. Moss, Aditya Ravuri, Sang Truong, Samuel Stanton, Gary Tom, Bojana Rankovic, Yuanqi Du, Arian Jamasb, Aryan Deshwal, Julius Schwartz, Austin Tripp, Gregory Kell, Simon Frieder, Anthony Bourached, Alex Chan, Jacob Moss, Chengzhi Guo, Johannes Durholt, Saudamini Chaurasia, Felix Strieth-Kalthoff, Alpha A. Lee, Bingqing Cheng, Alán Aspuru-Guzik, Philippe Schwaller, and Jian Tang. Gauche: A library for gaussian processes in chemistry, 2023.

[57] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *CoRR*, abs/1611.08402, 2016. URL http://arxiv.org/abs/1611.08402.

[58] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pages 488–495. PMLR, 2009.

[59] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[60] Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations, 2019.

[61] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks, 2022.

[62] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks, 2020.

[63] Yuanqing Wang and Theofanis Karaletsos. Stochastic aggregation in graph neural networks. *arXiv preprint arXiv:2102.12648*, 2021.

[64] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy,

Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL http://arxiv.org/abs/1912.01703.

[65] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22092–22103. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb4c835feb0a65cc39739320d7a51c02-Paper.pdf.

[66] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. A survey on graph kernels. *CoRR*, abs/1903.11835, 2019. URL http://arxiv.org/abs/1903.11835.

[67] Yuanqing Wang, Josh Fass, Benjamin Kaminow, John E Herr, Dominic Rufa, Ivy Zhang, Iván Pulido, Mike Henry, Hannah E Bruce Macdonald, Kenichiro Takaba, et al. End-to-end differentiable construction of molecular mechanics force fields. *Chemical Science*, 13(41): 12016–12033, 2022.

[68] Kenichiro Takaba, Iván Pulido, Mike Henry, Hugo MacDermott-Opeskin, John D Chodera, and Yuanqing Wang. Espaloma-0.3. 0: Machine-learned molecular mechanics force field for the simulation of protein-ligand systems and beyond. *arXiv preprint arXiv:2307.07085*, 2023.

[69] Michael Retchin, Yuanqing Wang, Kenichiro Takaba, and John D. Chodera. Druggym: A testbed for the economics of autonomous drug discovery. *bioRxiv*, 2024. doi: 10.1101/2024.05.28.596296. URL https://www.biorxiv.org/content/early/2024/06/02/2024.05.28.596296.

[70] Yuanqing Wang, Iván Pulido, Kenichiro Takaba, Benjamin Kaminow, Jenke Scheen, Lily Wang, and John D Chodera. Espalomacharge: Machine learning-enabled ultrafast partial charge assignment. *The Journal of Physical Chemistry A*, 128(20):4160–4167, 2024.

[71] Yuanqing Wang. *Graph Machine Learning for (Bio)Molecular Modeling and Force Field Construction*. PhD thesis, 2023. URL http://proxy.library.nyu.edu/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fdissertations-theses%2Fgraph-machine-learning-bio-molecular-modeling%2Fdocview%2F2789704784%2Fse-2%3Faccountid%3D12768. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2024-04-24.

[72] Yuanqing Wang and John D Chodera. Spatial attention kinetic networks with e (n)-equivariance. In *ICLR 2023*, 2023.

[73] Justin Grimmer, Margaret E Roberts, and Brandon M Stewart. Machine learning for social science: An agnostic approach. *Annual Review of Political Science*, 24:395–419, 2021.

[74] Gerhard Hessler and Karl-Heinz Baringhaus. Artificial intelligence in drug design. *Molecules*, 23(10):2520, 2018.

[75] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[76] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017.

[77] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications, 2018.

[78] Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. In *Conference on neural information processing systems*, pages 1–8, 2018.

[79] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2013.

[80] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[81] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. *CoRR*, abs/1905.07953, 2019. URL http://arxiv.org/abs/1905.07953.

[82] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. Graphsaint: Graph sampling based inductive learning method. *CoRR*, abs/1907.04931, 2019. URL http://arxiv.org/abs/1907.04931.

[83] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. *CoRR*, abs/1801.10247, 2018. URL http://arxiv.org/abs/1801.10247.

[84] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-dependent importance sampling for training deep and large graph convolutional networks. *CoRR*, abs/1911.07323, 2019. URL http://arxiv.org/abs/1911.07323.

[85] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks, 2020. URL https://arxiv.org/abs/2004.11198.

[86] Chuxiong Sun, Hongming Gu, and Jie Hu. Scalable and adaptive graph neural networks with self-label-enhanced training, 2021.

[87] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S. Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. *ArXiv*, abs/1901.01484, 2019. URL https://api.semanticscholar.org/CorpusID:57573752.

[88] Vijay Lingam, Chanakya Ekbote, Manan Sharma, Rahul Ragesh, Arun Iyer, and Sundararajan Sellamanickam. A piece-wise polynomial filtering approach for graph neural networks, 2021.

[89] Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers, 2023.

Disclosures. YW has limited financial interests in Flagship Pioneering, Inc. and its subsidiaries.

Funding. YW acknowledges support from the Schmidt Science Fellowship, in partnership with the Rhodes Trust, and the Simons Center for Computational Physical Chemistry at New York University.

A Experimental details

Code availability. All architectures, as well as scripts to execute the experiment, are distributed open-source under MIT license at `https://anonymous.4open.science/r/rum-834D/`. Core dependencies of our package include PyTorch [64] and Deep Graph Library [38].

Hyperparameters. All models are optimized using Adam [75] optimizer and SiLU [76] activation functions. 4 random walk samples are drawn everywhere unless specified. Other hyperparameters—learning rate ($10^{-5} \sim 10^{-2}$), hidden dimension ($32 \sim 64$), L2 regularization strength ($10^{-8} \sim 10^{-2}$), walk length ($3 \sim 16$), temperature for $\mathcal{L}_{\texttt{consistency}}$ ($0 \sim 1$), coefficient for $\mathcal{L}_{\texttt{consistency}}$ ($0 \sim 1$), coefficient for $\mathcal{L}_{\texttt{self}}$, and dropout probability—are tuned using the Ray platform [77] with the default Ax [78] search algorithm with 1000 trails or 24 hours tuning budget on a Nvidia A100® GPU.

B Additional technical details

B.1 Self-supervised regularization.

The stochasticity encoded in our model naturally affords it with some level of regularization. Apart from using the consistency loss ($\mathcal{L}_{\texttt{consistency}}$) used in Feng et al. [65] for classifications, we further regularize the model by using the RNNs in ϕ_x to predict the semantic representation of the next node on the walk given ω_u and jointly maximize this likelihood:

$$\hat{\omega}_{x_{i+1}} = g(\{\omega_{x_1}, \omega_{x_2}, \ldots, \omega_{x_i}\}, \omega_u | \theta); \tag{13}$$

$$\mathcal{L}_{\texttt{self}}(\theta) = -\log P(\hat{\omega}_{x_{i+1}} | \theta), \tag{14}$$

where $g(\cdot|\theta)$ is modeled as the sequence output of the RNN ϕ_x in Equation 5. The total loss is modeled as a combination of:

$$\mathcal{L}(\theta) = -\log P(y|\mathcal{G}, \mathbf{X}, \theta) + \mathcal{L}_{\texttt{self}} + \mathcal{L}_{\texttt{consistency}} \tag{15}$$

B.2 RUM attenuates over-squashing

Similarly, we can show that if the composing neural networks defy the vanishing gradient problem (w.r.t. the input) [79], the *sensitivity analysis* in Equation 3 [15] has a lower bound for RUM.

Lemma 7 (RUM attenuates over-squashing). *If ϕ_x, f have lower-bounded derivatives, the inter-node Jacobian for nodes u, v separated by a shortest path of length l, RUM with walk length l also has a lower bound:*

$$|\frac{\partial \mathbf{X}_v^{(l)}}{\partial \mathbf{X}_u^{(0)}}| \geq |\nabla \phi_x||\nabla f|(\hat{A}^l)_{uv}, \tag{16}$$

where $\hat{A}_{ij} = A_{ij} / \sum_j A_{ij}$ is the degree-normalized adjacency matrix.

Like the upper bound in Equation 3, this lower bound is also controlled by the power of the (normalzied) adjacency matrix, albeit the absence of self-loop will result in a slightly looser bottleneck. The term $(\hat{A}^l)_{uv}$ corresponds to the probability of the shortest path among all possible walks as a product of inverse node degrees (see Equation 4). There is no denying that the lower bound is still controlled by the power of the adjacency matrix, which corresponds to the exponentially growing receptive fields. One can also argue that, without prior knowledge, the contribution of the sensitivity analysis by the power of the graph adjacency matrix can never be alleviated, since there are always roughly $1/(\hat{A}^{l+1})_{uv}$ (assuming uniform node degree) structurally equivalent nodes. Nevertheless, since ϕ_x is not necessarily an iterative function, we alleviate the over-squashing problem by eliminating the power of the update function gradient term.

Now, we plug in the layer choices of ϕ_x—a GRU [16] unit. Its success, just like that of long short-term memory (LSTM) [80], can be attributed to the near-linear functional form of the long-range gradient. The term $|\nabla \phi_x|$ is controlled by a sequence of sigmoidal update gates, which can be optimized to approach 1 (fully open). If we ignore the gradient contribution of $\mathbf{X}_u^0$ to these gates, the non-linear activation function has only been applied exactly *once* on $\mathbf{X}_u^0$; therefore, the gradient $|\partial \mathbf{X}_v^{(l+1)} / \partial \mathbf{X}_u^{(0)}|$ is neither rapidly vanishing nor exploding.

$\omega_u = \mathbf{0}$	$\omega_x = \mathbf{0}$	$\mathcal{L}_{\texttt{self}} = 0$	$\mathcal{L}_{\texttt{consistency}} = 0$
82.2 ± 1.0	35.0 ± 1.0	78.4 ± 0.1	80.3 ± 1.1

Table 6: **Ablation study.** Cora [47] test set accuracy ↑ with in the architecture deleted.

B.3 Long-range neighborhood matching (Q3).

Once identifying the target leaf, this task seems trivial; nonetheless, this piece of information needs to be passed through layers of aggregation and non-linear update and is usually lost in the convolution. Since, on this binary tree, the receptive field grows exponentially, Alon and Yahav [14] argues that there is a theoretical lower boundary for the layer width D for the convolutional GNN to be able to encode all possible combinations of leaves, which is 2^{32D} for single-precision floating point (`float32`). This corresponds to the structural $\hat{A}^l$ term in Equation 3 and Equation 16. Evidently, when $D = 32$, as is the adopted experimental setting in Figure 5, this limit is far from being hit. So we hypothesize that the reason why convolutional GNNs cannot overfit the training set is because of the limitation of the functional forms, which are remedied by RUM, which shows 100% accuracy up to tree depth or problem radius $r = 5$, and a relatively moderate decrease afterward. Note that when the problem radius exceeds $r = 7$, RUM's performance is not significantly different from the convolutional counterparts.

B.4 Scalaing to large graphs (Q6).

In Appendix Table 8, we apply RUM on an ultra-large graph `OGB-PRODUCTS`, that cannot fit easily on a single GPU, and compare RUM with architectures specifically designed for large graphs [41].

B.5 Ablation study (Q7).

In Table 6, we conduct a brief ablation study where we delete, one by one, the components introduced in § 3. $\omega_u = \mathbf{0}$ and $\omega_x = \mathbf{0}$ refer to the deletion of the topological and semantic representations of walks, respectively. Neglecting topological information results in a moderate decrease in performance, whereas neglecting semantic representation is more detrimental. The $\omega_u = \mathbf{0}$ also resembles Jin et al. [17] albeit with different walk-wise aggregation. This offers a qualitative comparison between our work and Jin et al. [17] as no source code was released for this package so no rigorous comparison was possible. We also see that the regularization methods are helpful to the performance, with self-supervision being more crucial. We attribute this effect to firstly the relative simplicity of the Cora classification task, and secondly the flexibility of the (overparametrized) RNNs.

C Additional results

Algorithm 1 anonymous experiment

```
  **Input:** $w = (v_0, v_1, \ldots, v_l)$
  $C \leftarrow 0; \Omega \leftarrow \text{Dict}(\{\})$
  **for** $v_i$ in $w$ **do**
      **If** $v_i$ in $\Omega$: $u_i \leftarrow \Omega[v_i]$; **Else** $\Omega[v_i] \leftarrow C; C \leftarrow C + 1$
  **end for**
  **Return:** $\omega_u = (u_i) = (u_0, u_1, \ldots, u_l)$
```

Remark 8 (Inequality in distribution). For two nodes v_1, v_2 with distribution of random walks terminating at v_1, v_2 not equal in distribution $p(\omega_u(w_1)) \neq p(\omega_u(w_2))$ or $p(\omega_x(w_1)) \neq p(\omega_x(w_2))$, the node representations in Equation 6 are also different $\psi(v_1) \neq \psi(v_2)$.

One way to construct h function in Equation 5 is to have $h(w)$ positive only where $p(\omega_u(w_1)) > p(\omega_u(w_2))$; the same thing can be argued for ω_x. In other words, one only needs to prove two walks terminating at two nodes $p(\omega_u(w_1)) \neq p(\omega_u(w_2))$ or $p(\omega_x(w_1)) \neq p(\omega_x(w_2))$ are not *equal in distribution* to verify that RUM can distinguish two nodes. Conversely, we can also show that $p(\omega_u(w_1)) = p(\omega_u(w_2))$ implies that v_1, v_2 are isomorphic without labels— we refer the readers to the **Theorem 1** in Micali and Zhu [39] for a discussion on reconstructing unlabelled *graphs* using anonymous experiments.

	Complexity	CSL accuracy
GCN[1]	$\mathcal{O}(N)$	10.0 ± 0.0
GAT[6]	$\mathcal{O}(N)$	10.0 ± 0.0
GIN[2]	$\mathcal{O}(N)$	10.0 ± 0.0
GraphSAGE[4]	$\mathcal{O}(N)$	10.0 ± 0.0
3WLGNN[62]	$\mathcal{O}(N^2)$	95.7 ± 14.8
RUM	$\mathcal{O}(N)$	93.2 ± 0.8

Table 7: Graph classification accuracy ↑ on CSL [60] synthetic dataset for graph isomorphism test.

	Accuracy	Memory (MB)	Throughput(iter/s)
GraphSAGE [4]	80.61 ± 0.16	415.94	37.69
ClusterGCN [81]	78.62 ± 0.61	10.62	156.01
GraphSAINT [82]	75.36 ± 0.34	10.95	143.51
FastGCN [83]	73.46 ± 0.20	11.54	93.05
LADIES [84]	73.51 ± 0.56	20.33	93.47
SGC [7]	67.48 ± 0.11	0.01	267.31
SIGN [85]	76.85 ± 0.56	16.21	208.52
SAGN [86]	81.21 ± 0.07	71.81	80.04
RUM	76.1 ± 0.50	47.64	119.93
w/o walks		47.56	139.45
only walks		0.03	950.66

Table 8: **Node classification accuracy and efficiency** on `OGB-PRODUCTS` [55]

	# params	Cora	Photo
GCN [1]	48K	87.14 ± 1.01	88.26 ± 0.83
GAT [6]	49K	88.03 ± 0.79	90.04 ± 0.68
GCNII [37]	49K	88.46 ± 0.82	89.94 ± 0.31
RAW-GNN		87.85 ± 1.52	
LanczosNet [87]	50K	87.77 ± 1.45	93.21 ± 0.85
GPR-GNN [31]	48K	88.57 ± 0.69	93.85 ± 0.28
PP-GNN [88]		89.52 ± 0.85	92.89 ± 0.37
Transformer	37K	71.83 ± 1.68	90.05 ± 1.50
Graphomer [27]	139K	67.71 ± 0.78	95.20 ± 4.12
Specformer [89]	32K	88.57 ± 1.01	95.48 ± 0.32
RUM	23K +20K initial proj.	89.01 ± 1.40	95.35 ± 0.26

Table 9: **Node classification** test accuracy ↑ and standard deviation with 60:20:20 random split.

D Missing mathematical arguments.

D.1 Examples of Theorem 4

Example 8.1 (Cycle detection.)**.** *A k-cycle $\mathcal{C}_k$ is a subgraph of $\mathcal{G}$ consisting of k nodes, each with degree two. The existence of k-cycle can be determined by:*

$$\mathbb{1}(\mathcal{C}_k \subseteq \mathcal{G}) = \mathbb{1}[P(\omega_{x_{j+1}} = \omega_{x_0}, \omega_{x_i} \neq \omega_{x_j}, \forall i < j) > 0] \tag{17}$$

Example 8.2 (Diameter.)**.** *The diameter, $\delta_{\mathcal{G}}$ of graph $\mathcal{G}$ which equals the length of the longest shortes path in $\mathcal{G}$, can be expressed as*

$$\delta_{\mathcal{G}} = \underset{l=|\omega_x|, \omega_{x_i} \neq \omega_{x_j}, \forall i \neq j}{\text{argmax}} |\omega_x| \tag{18}$$

D.2 Proof of Theorem 4

Proof. First, we enumerate all possible *unlabelled* graphs with three nodes satisfying Assumption 3—one with two edges, one with three edges. (Note that there is only one non-isomorphic graph with two nodes.) Now we consider random walks of length $l = 3$, where

$$P(\omega_{u_3} = \omega_{u_0}) > 0 \tag{19}$$

only stands for the graph with three edges, but not with two edges, just like Example 8.1.

Furthermore, we can also distinguish between the 2-degree node and the 1-degree node in the graph with 2 edges and 3 nodes simply by verifying that

$$P(\omega_{u_1} \neq \omega_{u_3}) > 0 \tag{20}$$

only stands when v_2 is the 2-degree node.

Moving on to the labeled 3-node graph case, we can reduce the problem to investigate whether RUM can distinguish 3-node graphs that are isomorphic when unlabeled, but non-isomorphic when labeled. For the three-edged graph, ω_x uniformly samples the labels of three nodes. For the two-edged graph, suppose the node labels are A, B, C, and we start from the node with 2 degrees B (with nodes bearing A and C labels locally, structurally isomorphic),

$$P(B|\omega_x(w_t)) = \begin{cases} 1, t = 2n, n \in \mathbb{N}, \\ 0, t = 2n+1, n \in \mathbb{N} \end{cases} \tag{21}$$

$$P(A|\omega_x(w_t)) = P(C|\omega_x(w_t)) = \begin{cases} 0, t = 2n, n \in \mathbb{N}, \\ 1/2, t = 2n+1, n \in \mathbb{N} \end{cases} \tag{22}$$

If graphs have the same ω_x, they have the same B and the same or swapped A, C. As such, we have proven Theorem 4 for graphs with 3 nodes.

Now we prove that Theorem 4 stand for graphs of N nodes, they also stand for graphs of $N+1$ nodes, the Reconstruction Conjecture [44]

Suppose we have two non-isomorphic graphs with $N+1$ nodes $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same RUM embedding $\Psi_1 = \Psi_2$. We enumerate all $N+1$ subgraphs with each node deleted for each of these two graphs. By the Reconstruction Conjecture, at least one pair of subgraphs are non-isomorphic. For this pair, suppose the deleted vertex is v (ruling out the trivial case where the label or connectivity of v is different for these two graphs), and two remaining subgraphs $\mathcal{G}_1^{\setminus v}, \mathcal{G}_2^{\setminus v}$; since ϕ_x, ϕ_u, f are injective, $\Psi_1 = \Psi_2$ implies ω_u, ω_x are *equal in distribution* for $\mathcal{G}_1, \mathcal{G}_2$. As such, the walk distribution

$$P(\omega_x(w), \omega_u(w)|w_0 = v, w_i \neq v, i > 0) \tag{23}$$

$$= P(\omega_x(w_0), \omega_u(w_0)|w_0 = v)P(\omega_x(w_{1\ldots}), \omega_u(w_{1\ldots})|\omega_x(w_{1\ldots}), \omega_u(w_{1\ldots}), w_i \neq v, i > 0) \tag{24}$$

are also *equal in distribution* for $\mathcal{G}_1, \mathcal{G}_2$.

If there is a link between v and the nodes in $\mathcal{G}_1, \mathcal{G}_2$, we assign a new label to contain both the old label and the connection. As such, if the second term is not equal in distribution for $\mathcal{G}_1, \mathcal{G}_2$, we would have

$$\Psi(\mathcal{G}_1^{\setminus v}) \neq \Psi \mathcal{G}_2^{\setminus v}), \tag{25}$$

which is in conflict with the assumption that Theorem 4 stands for graphs with N nodes. □

D.3 Proof of Lemma 6

Proof. By the definition of Dirichlet energy (Equation 2) and non-contractive mappings (Definition 5),

$$\mathcal{E}(\psi(\mathbf{X})) = \frac{1}{N} \sum_{u,v \in \mathcal{E}_\mathcal{G}} ||\psi(\omega_x(\mathbf{X}_u)) - \psi(\omega_x(\mathbf{X}_v))||^2 \tag{26}$$

$$= \frac{1}{N} \sum_{u,v \in \mathcal{E}_\mathcal{G}} ||f(\phi_x(\omega_x(\mathbf{X}_u))) - f(\phi_x(\omega_x(\mathbf{X}_v)))||^2 \tag{27}$$

$$\geq \frac{1}{N} \sum_{u,v \in \mathcal{E}_\mathcal{G}} ||\phi_x(\omega_x(\mathbf{X}_u)) - \phi_x(\omega_x(\mathbf{X}_v))||^2 \tag{28}$$

$$= \frac{1}{N} \sum_{u,v \in \mathcal{E}_\mathcal{G}} ||\phi_x(\sum_{\{w\},|w|=l,w_l=u} p(w)\mathbf{X}_u, u \in w) - \phi_x(\sum_{\{w\},|w|=l,w_l=v} p(w)\mathbf{X}_v, v \in w)||^2 \tag{29}$$

$$\geq \frac{1}{N} \sum_{u,v \in \mathcal{E}_\mathcal{G}} ||(\sum_{\{w\},|w|=l,w_l=u} p(w)\mathbf{X}_u, u \in w) - (\sum_{\{w\},|w|=l,w_l=v} p(w)\mathbf{X}_v, v \in w)||^2 \tag{30}$$

$$\geq \mathcal{E}(\mathbf{X}) \tag{31}$$

The last inequality is due to the fact that ϕ_x is non-contractive w.r.t. all, and therefore, the last element of the sequence, which are always $\mathbf{X}_u$ and $\mathbf{X}_v$. □

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have claimed the theoretical and experimental advantage of RUM, which have been illustrated in § 4 and § 5, respectively.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the **Limitations** paragraph in § 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The assumptions are presented in Assumpsions 3, 2. The proofs are given in the Appendix § D.2.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The details are given in Appendix § A with the code for all models and experiments given in `https://anonymous.4open.science/r/rum-834D`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is open source at `https://anonymous.4open.science/r/rum-834D`. All data is from public benchmark datasets, as discussed in § 5.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The experimental details are described in Appendix § A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All error bars of real-world benchmarks are included in the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The type of compute is detailed in Appendix § A. See § 4 for a discussion on the speed and resource requirements of our model.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The Code of Ethics have been closely followed in every step of this research project.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See the **Social Impact** paragraph in § 6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets used in this work is open-source and has been properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human subjects are involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not pose risks incurred by study participants.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.