AN ANALYSIS OF ATTENTIVE WALK-AGGREGATING GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) have been shown to possess strong representation power, which can be exploited for downstream prediction tasks on graph-structured data, such as molecules and social networks. They typically learn representations by aggregating information from the K-hop neighborhood of individual vertices or from the enumerated walks in the graph. Prior studies have demonstrated the effectiveness of incorporating weighting schemes into GNNs; however, this has been primarily limited to K-hop neighborhood GNNs so far. In this paper, we aim to extensively analyze the effect of incorporating weighting schemes into walk-aggregating GNNs. Towards this objective, we propose a novel GNN model, called AWARE, that aggregates information about the walks in the graph using attention schemes in a principled way to obtain an end-to-end supervised learning method for graph-level prediction tasks. We perform theoretical, empirical, and interpretability analyses of AWARE. Our theoretical analysis provides the first provable guarantees for weighted GNNs, demonstrating how the graph information is encoded in the representation, and how the weighting schemes in AWARE affect the representation and learning performance. We empirically demonstrate the superiority of AWARE over prior baselines in the domains of molecular property prediction (61 tasks) and social networks (4 tasks). Our interpretation study illustrates that AWARE can successfully learn to capture the important substructures of the input graph.

1 INTRODUCTION

The increasing prominence of ML applications for graph-structured data has lead to the popularity of graph neural networks (GNNs) in several domains, such as social networks (Kipf & Welling, 2016), molecular property prediction (Duvenaud et al., 2015), and recommendation systems (Ying et al., 2018). Several empirical and theoretical studies (e.g., (Duvenaud et al., 2015; Kipf & Welling, 2016; Xu et al., 2018; Dehmamy et al., 2019)) have shown that GNNs can achieve strong representation power by constructing representations encoding rich information about the graph.

The most popular approach of learning GNNs involves aggregating information from the K-hop neighborhood of individual vertices in the graph (e.g., (Kipf & Welling, 2016; Gilmer et al., 2017; Xu et al., 2018)). An alternative approach for learning graph representations is via *walk aggregation* (e.g., (Vishwanathan et al., 2010; Shervashidze et al., 2011; Perozzi et al., 2014)) by means of using information of the enumerated walks in the graph. Existing studies have shown that walk-aggregating GNNs can achieve strong empirical performance with concrete analysis of the encoded graph information (Liu et al., 2019). This can potentially allow emphasizing and aggregating important walks, which can alleviate the problem of over-squashing exponentially growing information for distant dependencies (Alon & Yahav, 2020).

It is important to note that the strong representation power of GNNs may not always translate to learning the best representation amongst all possible ones. While this allows encoding all kinds of information, a subset of the encoded information that is not relevant for prediction may interfere or even overwhelm the information useful for learning—leading to sub-optimal performance. A particularly attractive approach to address this challenge is by incorporating weighting schemes into GNNs, which is inspired by the powerful empirical performance of attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015; Xu et al., 2015; Vaswani et al., 2017) for natural language processing (e.g., (Devlin et al., 2019)) and computer vision tasks (e.g., (Dosovitskiy et al., 2020)).

In the domain of graph representation learning, recent studies (Gilmer et al., 2017; Veličković et al., 2017; Yun et al., 2019; Maziarka et al., 2020; Rong et al., 2020) have used the self-attention mechanism to improve the empirical performance of GNNs by learning to select important information and removing the irrelevant ones. These studies; however, have only explored using attention schemes for K-hop neighborhood GNNs, and there has been no corresponding work exploring this idea for walk-aggregating GNNs. Incorporating attention mechanisms into walk-aggregating GNNs can allow for a finer-grained analysis of the weighting schemes at different granularities of the graph components. Furthermore, a majority of these prior studies have been empirically driven—lacking a strong theoretical understanding about success conditions.

In this paper, we propose to theoretically and empirically examine the effect of incorporating weighting schemes into walk-aggregating GNNs. To this end, we propose a simple, interpretable, and end-to-end supervised GNN model, called AWARE (**A**ttentive **W**alk-**A**ggregating G**R**aph Neural N**E**twork), for graph-level prediction. AWARE aggregates the walk information by means of weighting schemes at distinct levels (vertex-, walk-, and graph-level) in a principled manner. By virtue of the incorporated weighting schemes at these different levels, AWARE can emphasize the information important for prediction while diminishing the irrelevant ones—leading to representations that can improve learning performance. We perform an extensive three-fold analysis of AWARE as summarized below:

- **Theoretical Analysis:** We present *provable guarantees* for AWARE, identifying conditions when the weighting schemes improve learning. Prior weighted GNNs (e.g., (Veličković et al., 2017; Maziarka et al., 2020)) do not enjoy similar theoretical guarantees, making this *the first provable guarantee on the learning performance of weighted GNNs* (to the best of our knowledge). Furthermore, current understanding of weighted GNNs typically focuses only on the positive effect of weighting on their representation power. In contrast, we also explore the limitation scenarios when the weighting does not translate to stronger learning power.
- **Empirical Analysis:** We empirically evaluate the performance of AWARE on graph-level prediction tasks from two domains: molecular property prediction (61 tasks from 11 popular benchmarks) and social networks (4 tasks). For both domains, AWARE overall outperforms both traditional graph representation methods as well as recent GNNs (including the ones that use attention mechanisms) in the standard setting (as defined in Section 3).
- **Interpretability Analysis:** We perform an interpretation study to support our design for AWARE as well as the theoretical insights obtained about the weighting schemes. We provide a visual illustration that AWARE can extract the important sub-graphs for the prediction tasks. Furthermore, we show that the weighting scheme in AWARE can align well with the downstream predictors.

2 Related Work

Graph neural networks (GNNs). GNNs have been the predominant method for capturing information of graph data (Li et al., 2015; Duvenaud et al., 2015; Kipf & Welling, 2016; Kearnes et al., 2016; Gilmer et al., 2017). A majority of GNN methods build graph representations by aggregating information from the K-hop neighborhood of individual vertices (Duvenaud et al., 2015; Li et al., 2015; Battaglia et al., 2016; Kearnes et al., 2016; Xu et al., 2018; Yang et al., 2019). This is achieved by maintaining a latent representation for every vertex, and iteratively updating it to capture information from neighboring vertices that are K-hops away. Another mainstream approach is enumerating the walks in the graph and using their information (Vishwanathan et al., 2010; Shervashidze et al., 2011; Perozzi et al., 2014). Liu et al. (2019) use the motivation of aggregating information from the walks by proposing a GNN model that can achieve strong empirical performance along with concrete theoretical analysis. Furthermore, properly aggregating important walk information can potentially alleviate the problem of over-squashing exponentially growing information for distant dependencies in K-hop neighborhood GNNs as discussed by Alon & Yahav (2020).

Theoretical studies have shown that GNNs have strong representation power (Xu et al., 2018; Dehmamy et al., 2019; Liu et al., 2019), and have inspired new disciplines for improving their representations further (Morris et al., 2019; Azizian & marc lelarge, 2021). To this extent, while the *standard setting* of GNNs has only vertex features and adjacency information as inputs, many recent GNNs (Kearnes et al., 2016; Gilmer et al., 2017; Coors et al., 2018; Yang et al., 2019; Klicpera et al., 2020; Wang et al., 2021) exploit extra information, such as edge features, to gain stronger

performance. In this work; however, we focus on *analyzing* the effect of applying attention schemes for representation learning, and thus want to perform this analysis under the standard setting first. In the future, one may possibly incorporate the aforementioned extra information (e.g., edge attributes and 3D spatial orientation) into our analysis.

GNNs with attention. The empirical effectiveness of attention mechanisms have been demonstrated on language (Martins & Astudillo, 2016; Devlin et al., 2019; Raffel et al., 2020) and vision tasks (Ramachandran et al., 2019; Dosovitskiy et al., 2020; Zhao et al., 2020). This has also been extended to the K-hop GNN research line where the main motivation is to dynamically learn a weighting scheme at various granularities, e.g., vertex- and graph-level. Graph Attention Network (GAT) (Veličković et al., 2017) and Molecule Attention Transformer (MAT) (Maziarka et al., 2020) utilize the attention idea in their message passing functions. Rong et al. (2020) applies an attention mechanism at both vertex- and edge-levels to better capture the structural information in molecules. ENN-S2S (Gilmer et al., 2017) adopts an attention module (Vinyals et al., 2015) as a readout function. However, all such studies are based on K-hop GNNs, and to the best of our knowledge, our work is the *first* to bring attention schemes into walk-aggregation GNNs.

3 PRELIMINARIES

Graph data. We assume an input graph $\mathcal{G}=(\mathcal{V},\mathcal{A})$ consisting of vertex attributes $\mathcal{V}$ and an adjacency matrix $\mathcal{A}$. We index the vertices by $[m]=\{1,\dots,m\}$. Suppose each vertex has C discrete-valued attributes, and the j^{th} attribute takes values in a set of size k_j. Let $h_i^j \in \{0,1\}^{k_j}$ be the one-hot encoding of the j^{th} attribute for vertex i. Then, the input of vertex i is the concatenation of C attributes, i.e., $h_i=[h_i^1;\dots;h_i^C] \in \{0,1\}^K$ where $K=\sum_{j=1}^{C} k_j$. Then $\mathcal{V}$ is the set $\{h_i\}_{i=1}^m$.

We denote the adjacency matrix by $\mathcal{A}\in\{0,1\}^{m\times m}$, where $\mathcal{A}_{i,j}=1$ indicates that vertices i and j are connected. We denote the set containing the neighbors of vertex i by $\mathcal{N}(i)=\{j\in[m]:\mathcal{A}_{i,j}=1\}$. For molecular graphs, vertices and edges correspond to atoms and bonds, respectively. For social network graphs, they correspond to entities (actors, online posts) and the connections between them. Although many GNNs exploit extra information like edge attributes, our primary focus is on the effect of attention mechanisms. Hence, we perform our analysis in the *standard setting* (i.e., only using vertex attributes and adjacency information).

Vertex embedding. We define an r-dimensional embedding of vertex i by:

$$f_i = W h_i, \tag{1}$$

where $W=[W^1;\dots;W^C]\in\mathbb{R}^{r\times K}$ and $W^j\in\mathbb{R}^{r\times k_j}$ is the embedding matrix for each attribute $j\in[C]$. We denote the embedding corresponding to $\mathcal{V}$ by $F=[f_1;\dots;f_m]$.

Walk aggregation. Unlike the typical approach of aggregating K-hop neighborhood information, walk aggregation enumerates the walks in the graph, and use their information (e.g., (Vishwanathan et al., 2010; Perozzi et al., 2014)). Liu et al. (2019) utilize the walk-aggregation strategy by proposing the N-gram graph GNN, which can achieve strong empirical performance, allow for fine-grained theoretical analysis, and potentially alleviate the over-squashing problem in K-hop GNNs. The N-gram graph views the graph as a Bag-of-Walks. Specifically, embeddings are constructed for *all* walks in the graph through element-wise product of vertex embeddings along the walks. The embeddings for all walks of length n are aggregated (by summation) to obtain embeddings for the n-gram walk set. Finally, the graph embeddings are obtained by concatenation of all n-gram walk set embeddings. Compared to K-hop aggregation strategies, this formulation explicitly allows analyzing representations at different granularities of the graph: vertices, walks, and the entire graph. This provides motivation for capitalizing on the N-gram walk-aggregation strategy for incorporating and analyzing the effect of weighting schemes on walk-aggregation GNNs. The principled design facilitates theoretical analysis of conditions under which the weighting schemes may be beneficial. Thus, in this paper, we analyze the effect of incorporating attention weighting schemes on the N-gram walk-aggregation GNN.

4 AWARE: ATTENTIVE WALK-AGGREGATING GRAPH NEURAL NETWORK

We propose AWARE, an end-to-end fully supervised GNN for learning graph embeddings by aggregating information from walks with learned weighting schemes. Intuitively, not all walks in a graph

are equally important for downstream prediction tasks. AWARE incorporates an attention mechanism to assign different contributions to individual walks as well as assigns feature weightings at the vertex and graph embedding levels. These weights are learned in a *supervised* fashion for prediction. This enables AWARE to mitigate the shortcomings of its unweighted counterpart (Liu et al., 2019), which computes graph embeddings in an *unsupervised* manner only using the graph topology.

At a high level, AWARE first computes vertex embeddings F, and initializes a latent vertex representation $F_{(1)}$ by incorporating a feature weighting at the vertex level. It then iteratively updates the latent representation $F_{(n)}$ using attention at the walk level, before performing a weighted summarization at the graph level to obtain embeddings $f_{(n)}$ for walk sets of length n. The $f_{(n)}$'s are concatenated to produce the graph embedding $f_{[T]}(G)$ for the downstream task. We now provide more details.

Weighted vertex embedding. Intuitively, some directions in the vertex embedding space are likely to be more important for the downstream prediction task than others. In the extreme case, the prediction task may depend only on a subset of the vertex attributes (corresponding to some directions in the embedding space), while the rest may be inconsequential and hence should be ignored when constructing the graph embedding. AWARE weights different vertex features using $W_v \in \mathbb{R}^{r' \times r}$ by computing the initial latent vertex representation $F_{(1)}$ as:

$$F_{(1)} = \sigma(W_v F), F \text{ is computed using Eqn (1)}$$

where σ is a non-linear activation function, and r' is the dimension of the weighted vertex embedding.

Walk attention. AWARE computes embeddings corresponding to walks of length n in an iterative manner, and updates the latent vertex representations in each iteration using such walk embeddings. When aggregating the embedding of a walk, each vertex in the walk is bound to have a different contribution towards the downstream prediction task. For instance, in molecular property prediction, the existence of chemical bonds between certain types of atoms in the molecule may have more impact on the property to be predicted than others. To achieve this, in iteration n, AWARE updates the latent representations for vertex i from $[F_{(n-1)}]_i$ to $[F_{(n)}]_i$ by taking an element-wise product of $[F_{(n-1)}]_i$ with a *weighted* sum of the latent representation vectors of its neighbors $j \in \mathcal{N}(i)$. Such a weighted update of the latent representations implicitly assigns a different importance to each neighbor j for vertex i. Assuming that the importance of vertex j for vertex i depends on their latent representations, we consider a score function corresponding to the update from vertex j to i as:

$$\mathbf{S}_{ji} := S(f_j, f_i). \tag{2}$$

To allow different weights $[\mathbf{S}_{(n)}]_{ji}$ for different iterations n, we use the latent representations for vertices from the previous iteration $(n-1)$. In particular, we use the self-attention mechanism:

$$[Z_{(n)}]_{j \to i} = [F_{(n-1)}]_j^\top W_w [F_{(n-1)}]_i$$

where $[F_{(n-1)}]_i$ is the latent vector of vertex i at iteration $n-1$, and $W_w \in \mathbb{R}^{r' \times r'}$ is a parameter matrix that is learned. We then define the attention weighting matrix used at iteration n as:

$$[\mathbf{S}_{(n)}]_{ji} = \frac{e^{[Z_{(n)}]_{j \to i}}}{\sum_{k \in \mathcal{N}(i)} e^{[Z_{(n)}]_{k \to i}}} \tag{3}$$

Using this attention matrix $\mathbf{S}_n$, we perform the iterative update to the latent vertex representations via a weighted sum of the latent representation vectors of their neighbors:

$$F_{(n)} = \Big(F_{(n-1)} (\mathcal{A} \odot \mathbf{S}_n) \Big) \odot F_{(1)}$$

This update is simple and efficient, and automatically aggregates important information from the vertex neighbors for the downstream prediction task.

Weighted summarization. Since the downstream task may selectively prefer certain directions in the final graph embedding space, AWARE learns a weighting $W_g \in \mathbb{R}^{r' \times r'}$ to compute a *weighted* sum of latent vertex representations for obtaining walk set embeddings of length n as follows:

$$f_{(n)} = \sigma(W_g F_{(n)}) \mathbf{1}$$

where $\mathbf{1}$ denotes matrix of ones in $\mathbb{R}^{m \times m}$. Walk set embeddings up to length T are then concatenated to produce the graph embedding $f_{[T]}(G) = [f_{(1)}, \ldots, f_{(T)}]$.

End-to-end supervised training. We summarize the different weighting schemes and steps of AWARE as a pseudo-code in Algorithm 1. The graph embeddings produced by AWARE can be fed into any properly-chosen predictor h_θ parametrized by θ, so as to be trained end-to-end on labeled data. For a given loss function $\mathcal{L}$, and a labeled data set $\{(G_i, y_i)\}$ where G_i's are graphs and y_i's are their labels, AWARE can learn the parameters (W, W_v, W_w, W_g) and the predictor θ by optimizing the loss $\sum_i \mathcal{L}(y_i, h_\theta(f_{[T]}(G_i)))$.

Algorithm 1 AWARE (W, W_v, W_w, W_g)

Require: Graph $G{=}(\mathcal{V}, \mathcal{A})$, max walk length T

1: Compute vertex embeddings F by Eqn (1)
2: $F_{(1)} = \sigma(W_v F)$
3: **for** each $n \in [2, T]$ **do**
4: Compute $\mathbf{S}_n$ using Eqn (3)
5: $F_{(n)}{=}\Big(F_{(n-1)}(\mathcal{A} \odot \mathbf{S}_n)\Big) \odot F_{(1)}$
6: **end for**
7: Set $f_{(n)} := \sigma(W_g F_{(n)})\mathbf{1}$ for $1 \le n \le T$
8: Set $f_{[T]}(G) := [f_{(1)}; \ldots; f_{(T)}]$

Ensure: The graph embedding $f_{[T]}(G)$

The N-Gram walk aggregation strategy termed as the N-Gram Graph (Liu et al., 2019) operates in two steps: first to learn a graph embedding using the graph topology without any supervision, and then to use a predictor on the embedding for the downstream task. In contrast, AWARE is end-to-end fully supervised, and simultaneously learns the vertex/graph embeddings for the downstream task along with the weighting schemes to highlight the important information in the graph and suppress the irrelevant and/or harmful ones. Secondly, the weighting schemes of AWARE allow for the use of simple predictors over the graph embeddings (e.g. logistic regression or shallow fully-connected networks) for performing end-to-end supervised learning. In contrast, N-Gram Graph requires strong predictors such as XGBoost (with thousands of trees) to exploit the encoded information in the graph embedding.

5 Theoretical Analysis

For the design of our walk-aggregation GNN with weighting schemes, we are interested in the following two fundamental questions: *What representation can it obtain? Under what conditions can the weighting scheme improve the prediction performance?* This section analyzes the weighting $\mathbf{S}_{ij} = S(f_i, f_j)$ for these two questions.[1] We assume:

- $W_v = W_g = I$, the number of attributes is $C = 1$, and the activation is linear $\sigma(z) = z$.

Here, the assumptions simplify the notations, and allow focusing on the effect of $\mathbf{S}_{ij}$. Appendix A.3 presents an analysis for more general W_v, W_g and $C > 1$.

We will show that the weighting scheme can highlight important information, and reduce irrelevant information for the prediction, and thus improve learning. To this end, we first analyze what information can be encoded in our graph representation, and how they are weighted (Theorem 1). We then examine why the weighting can help learning a predictor with better performance (Theorem 2). We focus on the intuition and implications here, and present proofs and more discussion in Appendix A.

Weighted representation in AWARE. We first formally define the walk information in the graph, and a notion of walk weights (which will be shown to be exactly the weights induced by our method). Recall that we assume the number of attributes is $C = 1$. K is the number of possible attribute values, and the columns of $W \in \mathbb{R}^{r \times K}$ are embeddings for different attribute values u. Let $W(u)$ denote the column for value u, i.e., $W(u) = Wh(u)$ where $h(u)$ is the one-hot vector of u.

Definition 1 (Walk Statistics)**.** *A* walk type *of length n is a sequence of n attribute values $v = (v_1, v_2, \cdots, v_n)$. The* walk statistics *vector $c_{(n)}(G) \in \mathbb{R}^{K^n}$ is the histogram of all walk types of length n in the graph G, i.e., each entry is indexed by a walk type v and the entry value is the number of walks with the attribute values v in the graph. Furthermore, let $c_{[T]}(G)$ be the concatenation of $c_{(1)}(G), \ldots, c_{(T)}(G)$. When G is clear from the context, we write $c_{(n)}$ and $c_{[T]}$ for short.*

Definition 2 (Walk Weights)**.** *The weight of a walk type $v = (v_1, \ldots, v_n)$ is $\lambda(v) := \prod_{i=1}^{n-1} S(W(v_i), W(v_{i+1}))$ where $S(\cdot, \cdot)$ is the weight function in Eq equation 2.*

[1] For the other weighting schemes W_v and W_g, we know W_v weights the vertex embeddings f_i, and W_g weights the final embeddings $F_{(n)}$, emphasizing important directions in the corresponding space. If W_v has singular vector decomposition $W_v = U\Sigma V^\top$, then it will relatively emphasize the singular vector directions with large singular values. Similar for W_g. See Section 7 for some visualization.

We have the following theorem which shows that our representation is a linear mapping of the weighted walk statistics.

Theorem 1. *The embedding $f_{(n)}$ is a linear mapping of the walk statistics $c_{(n)}$:*

$$f_{(n)} = \mathcal{M}_{(n)}\Lambda_{(n)}c_{(n)}$$

where $\mathcal{M}_{(n)}$ is a matrix depending only on W, and $\Lambda_{(n)}$ is a K^n-dimensional diagonal matrix whose columns are indexed by walk types v, and have diagonal entries $\lambda(v)$. Therefore, $f_{[T]} := \mathcal{M}\Lambda c_{[T]}$ where $\mathcal{M}$ is a block-diagonal matrix with diagonal blocks $\mathcal{M}_{(1)}, \mathcal{M}_{(2)}, \ldots, \mathcal{M}_{(T)}$, and Λ is block-diagonal with blocks $\Lambda_{(1)}, \Lambda_{(2)}, \ldots, \Lambda_{(T)}$.

In words, $c_{(n)}$ is first weighted by our weighting scheme where the count of each walk type v is weighted by the corresponding walk weight $\lambda(v)$, and then compressed from the high dimension $\mathbb{R}^{K^n}$ to the low dimension $\mathbb{R}^r$. Ideally, we would like to have relatively larger weights on walk types important for the prediction task and smaller for those not important. This provides the basis for the focus of our analysis: the effect of weighting for the learning performance.

Learning guarantees of AWARE. To illustrate the effect of weighting, suppose the label is given by a linear function on $c_{[T]}$ with parameter β^*, i.e., $y = \langle \beta^*, c_{[T]} \rangle$. First, consider learning a linear function over the weighted features $\Lambda c_{[T]}$. If Λ is invertible, the parameter $\Lambda^{-1}\beta^*$ on $\Lambda c_{[T]}$ has the same output as β^* on $c_{[T]}$, and thus has the same loss. So, we only need to learn $\Lambda^{-1}\beta^*$. The sample size needed will depend on the factor $\|\Lambda^{-1}\beta^*\|_2 \|\Lambda c_{[T]}\|_2$, which is potentially smaller than $\|\beta^*\|_2 \|c_{[T]}\|_2$ for the unweighted case. This means fewer data samples are needed (equivalently, smaller loss for a fixed amount of samples).

Now, consider the case of learning over $f_{[T]}(G) = \mathcal{M}\Lambda c_{[T]}$ that has an extra $\mathcal{M}$ (defined in Theorem 1). We note that $c_{[T]}$ can be sparse compared to its high dimension (since only a very small fraction of all possible walk types will likely appear in a graph). Well-established results from compressive sensing show that when $\mathcal{M}$ has the Restricted Isometry Property (RIP), learning over $\mathcal{M}\Lambda c_{[T]}$ is comparable to learning over $\Lambda c_{[T]}$. Indeed, Theorem 4 in Appendix A shows when W is random and the embedding dimension r is large enough, there are families of distributions of W such that $\mathcal{M}$ has RIP for $\Lambda c_{[T]}$. Thus, we assume $\mathcal{M}$ has RIP, and focus on the analysis of how W_w affects the weighting and the learning.

However, the above intuition is only for learning over a *fixed* weighting Λ induced by a *fixed* W_w, while in fact W_w needs to be *learned*. Our key challenge is to incorporate the learning of W_w in the analysis. Formally, we consider learning W_w from a hypothesis class $\mathcal{W}$, and let $\Lambda(W_w)$ and $f_{[T]}(G; W_w)$ denote the weights and representation given by W_w. For prediction, we consider binary classification with the logistic loss $\ell(g, y) = \log(1 + \exp(-gy))$ where g is the prediction and y is the true label. Let $\ell_{\mathcal{D}}(\theta, W_w)$ be the risk of a linear classifier with a parameter θ on $f_{[T]}(G; W_w)$ over the data distribution $\mathcal{D}$. Given M i.i.d. samples $\{(G_i, y_i)\}_{i=1}^M$ from $\mathcal{D}$, consider learning over $W_w \in \mathcal{W}$, $\|\theta\|_2 \le B_\theta$ for a regularization coefficient B_θ:

$$\hat{\theta}, \widehat{W_w} = \arg\min \frac{1}{M} \sum_{i=1}^{M} \ell\Big(\langle \theta, f_{[T]}(G_i; W_w) \rangle, y_i\Big).$$

To derive error bounds, suppose $\mathcal{W}$ is equipped with a norm $\|\cdot\|$ and let $\mathcal{N}(\mathcal{W}, \epsilon)$ be the ϵ-covering number of $\mathcal{W}$ w.r.t. the norm $\|\cdot\|$ (other complexity measures on $\mathcal{W}$, such as VC-dimension, can also be used). Furthermore, let β^* denote the best linear classifier on $c_{[T]}$, and let $\ell^*_{\mathcal{D}}$ denote its risk.

Theorem 2. *Assume $c_{[T]}$ is s-sparse, $\mathcal{M}$ satisfies $(2s, \epsilon_0)$-RIP, $\Lambda(W_w)$ is invertible and $f_{[T]}(G; W_w)$ is L_f-Lipschitz over $\mathcal{W}$. For any $\delta, \epsilon \in (0, 1)$, there are regularization coefficient values B_θ such that with probability $\ge 1 - \delta$:*

$$\ell_{\mathcal{D}}(\hat{\theta}, \widehat{W_w}) \le \ell^*_{\mathcal{D}} + 2\epsilon + O\left(\sqrt{\frac{rT + \mathcal{C}_\epsilon(\mathcal{W})}{M}}\right) + \min_{W_w \in \mathcal{W}} B(W_w) \times O\left(\sqrt{\epsilon_0 + \frac{\mathcal{C}_\epsilon(\mathcal{W})}{M}}\right)$$

where $\mathcal{C}_\epsilon(\mathcal{W}) := \log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right) + \log \frac{1}{\delta}$, and

$$B(W_w) := \max_{G \sim \mathcal{D}} \|\Lambda(W_w) c_{[T]}(G)\|_2 \|\Lambda(W_w)^{-1}\beta^*\|_2.$$

Remark. Theorem 2 shows that the learned model has risk comparable to that of the best linear classifier on the walk statistics, given sufficient data. Let's now compare to the unweighted case (i.e., Λ is the identity matrix), where $\log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right)$ reduces to 0, and $B(W_w)$ reduces to $B_0 := \max_{G\sim\mathcal{D}} \|c_{[T]}(G)\|_2 \|\beta^*\|_2$. So, our method needs extra samples to learn W_w, leading to the extra error terms related to $\log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right)$. On the other hand, the benefit of weighting is replacing the factor B_0 with $\min_{W_w} B(W_w)$. If there is W_w^* with $B(W_w^*) \ll B_0$, the error is significantly reduced. Therefore, there is a trade-off between the reduction of error for learning classifiers on an appropriate weighted representation and the additional samples needed for learning an appropriate weighting. When the labels indeed depend on some important walks rather than on all walks, the weighting can alleviate the problem of over-squashing shown in Alon & Yahav (2020).

Our analysis also reveals that weighting is not panacea: its ability to improve learning depends on (1) whether it can assign the large weights to important features, and (2) *whether the benefit of weighting important features exceeds the overhead of learning a proper weighting scheme*. (1) is generally accepted while (2) is largely overlooked (at least not explicitly discussed) in existing studies.

An illustrative example. The benefit of weighting can be significant in practice. $\min_{W_w} B(W_w)$ can be much smaller than B_0, especially when some features (i.e., walk types) in $c_{[T]}$ are important while others are not, which is true for many real-world applications. Suppose $c_{[T]}(G)$ is s-sparse with each non-zero entry being some constant c. Suppose only a few of the features are useful for the prediction, i.e., β^* is ρ-sparse with each non-zero entry being some constant b, and $\rho \ll s$. Suppose there is a weighting W_w^* that leads to weight Υ on the entries corresponding to the ρ important features (i.e., the non-zero entries in β^*), and weight υ for the other features where $|\upsilon| \ll |\Upsilon|$. Then it can be shown that $\frac{\min_{W_w} B(W_w)}{B_0} \leq \sqrt{\frac{\rho}{s} + \left(1-\frac{\rho}{s}\right)\left(\frac{\upsilon}{\Upsilon}\right)^2}$. Since $\rho \ll s$ and $|\upsilon| \ll |\Upsilon|$, $\min_{W_w} B(W_w)$ is much smaller than B_0, so the weighting can significantly reduce the error.

This example demonstrates that with a proper weighting that highlights important features and depresses irrelevant ones for prediction, the error can be much smaller than the error without weighting. If this benefit exceeds the overhead in learning the proper weighting, the learning performance is improved. This trade-off is indeed visible on many datasets; see our experimental results in Section 6. The analysis also shows that a trained AWARE model finds a weighting that automatically highlights important substructures like edges in the graph and provide useful interpretation; see the interpretation study in Section 7.

6 Experiments

We evaluate AWARE on graph-level prediction tasks from two domains: molecular property prediction (61 tasks from 11 benchmarks) and social networks (4 benchmarks).[2] Specifically, we consider: 37 classification (33 molecular + 4 social networks) and 28 regression (on molecular) tasks in total. For more details about the datasets, see Appendix B.

Baseline methods. We consider WL kernels (Shervashidze et al., 2011), Morgan fingerprints (Morgan, 1965), and N-Gram Graph (Liu et al., 2019) as baselines for graph representation learning. For the predictor on top of the representations, we use SVM for WL kernels, and Random Forest and XGBoost (Chen & Guestrin, 2016) for Morgan fingerprints and N-Gram Graph. We also consider several recent end-to-end trainable GNNs that are commonly used, including GCNN (Duvenaud et al., 2015), GAT (Veličković et al., 2017), GIN (Xu et al., 2018), Attentive FP (Xiong et al., 2019), and PNA (Corso et al., 2020). Note that we do not consider recent GNN models that use extra edge/3D information or self-supervised pre-training as baselines to avoid unfair comparison to AWARE—since our analysis throughout this paper focuses on *the standard setting* (see Section 3). Attentive FP and PNA were run without using extra edge information as this is not their main contribution.

Evaluation. We perform *single-task learning* for each task in each dataset. Each dataset is randomly split into training, validation, and test sets with a ratio of 8:1:1, respectively. We report the average performance across 5 runs (datasets are split independently for each run). We select optimal hyperparameters using grid search. Full hyperparameter details as well as an ablation study on their effects are presented in Appendices D and F, respectively. For the molecular property prediction tasks, we

[2]Code has been submitted in the supplementary material, and will be made public upon acceptance.

Table 1: Overall performance on all 15 datasets (65 tasks). We report (# tasks with top-1 performance, # tasks with top-3 performance). Models with no top-3 performance on a dataset are left blank. Models that are too slow, not well tuned, or not run due to model/dataset incompatibility are marked with "–". For full results with error bounds, see Tables S6, S7, and S8 in the appendix.

Dataset	# Tasks	Metric	Morgan FP	WL Kernel	GCNN	GAT	GIN	Attentive FP	PNA	N-Gram Graph	AWARE
IMDB-BINARY	1	ACC	–					(0, 1)	(0, 1)		(1, 1)
IMDB-MULTI	1	ACC	–					(0, 1)	(0, 1)		(1, 1)
REDDIT-BINARY	1	ACC	–				(0, 1)		(0, 1)		(1, 1)
COLLAB	1	ACC	–				(0, 1)		(0, 1)		(1, 1)
MUTAGENICITY	1	ACC	–		(1, 1)				(0, 1)		(0, 1)
TOX21	12	ROC	(0, 4)	(0, 2)				(0, 5)	(1, 3)	(4, 11)	(7, 11)
CLINTOX	2	ROC				(1, 1)	(0, 1)	(0, 1)	(0, 1)		(1, 2)
HIV	1	ROC	(1, 1)				(0, 1)			(0, 1)	
MUV	17	ROC	(2, 7)	(3, 4)	(0, 8)	(0, 1)	(0, 3)	(1, 2)	(1, 6)	(1, 4)	(9, 16)
DELANEY	1	RMSE						(0, 1)		(0, 1)	(1, 1)
MALARIA	1	RMSE	(1, 1)						(0, 1)	(0, 1)	
CEP	1	RMSE					(1, 1)	(0, 1)	(0, 1)		
QM7	1	MAE						(0, 1)		(0, 1)	(1, 1)
QM8	12	MAE			(5, 6)		(1, 7)		(0, 1)	(0, 11)	(6, 11)
QM9	12	MAE		–	(3, 12)		(4, 7)			(1, 11)	(4, 6)
Total	65		(4, 13)	(3, 6)	(9, 27)	(1, 2)	(6, 22)	(1, 13)	(2, 18)	(6, 41)	**(33, 53)**

Table 2: Performance on several tasks from both molecular and social network domains. The top-3 and best performing models for each task are highlighted in gray and **blue**, respectively.

Task	Metric	Morgan FP	WL Kernel	GCNN	GAT	GIN	Attentive FP	PNA	N-Gram Graph	AWARE
IMDB-BINARY	ACC	–	$0.680_{\pm 0.022}$	$0.698_{\pm 0.026}$	$0.568_{\pm 0.047}$	$0.696_{\pm 0.037}$	$0.716_{\pm 0.022}$	$0.710_{\pm 0.011}$	$0.522_{\pm 0.036}$	**$0.740_{\pm 0.020}$**
REDDIT-BINARY	ACC	–	$0.892_{\pm 0.017}$	$0.931_{\pm 0.013}$	$0.900_{\pm 0.036}$	$0.933_{\pm 0.009}$	$0.864_{\pm 0.029}$	$0.938_{\pm 0.010}$	$0.764_{\pm 0.026}$	**$0.949_{\pm 0.014}$**
COLLAB	ACC	–	$0.567_{\pm 0.011}$	$0.660_{\pm 0.009}$	$0.616_{\pm 0.029}$	$0.669_{\pm 0.014}$	$0.653_{\pm 0.012}$	$0.675_{\pm 0.024}$	$0.376_{\pm 0.119}$	**$0.739_{\pm 0.017}$**
CT_TOX	ROC	$0.813_{\pm 0.036}$	$0.830_{\pm 0.057}$	$0.860_{\pm 0.027}$	$0.828_{\pm 0.075}$	$0.859_{\pm 0.063}$	$0.873_{\pm 0.053}$	$0.895_{\pm 0.043}$	$0.849_{\pm 0.024}$	**$0.905_{\pm 0.038}$**
FDA_APPROVED	ROC	$0.795_{\pm 0.084}$	$0.862_{\pm 0.029}$	$0.866_{\pm 0.028}$	**$0.899_{\pm 0.033}$**	$0.883_{\pm 0.025}$	$0.870_{\pm 0.070}$	$0.879_{\pm 0.022}$	$0.852_{\pm 0.044}$	$0.895_{\pm 0.050}$
DELANEY	RMSE	$1.081_{\pm 0.073}$	$1.160_{\pm 0.050}$	$0.762_{\pm 0.151}$	$0.954_{\pm 0.151}$	$0.840_{\pm 0.070}$	$0.615_{\pm 0.026}$	$0.922_{\pm 0.122}$	$0.744_{\pm 0.068}$	**$0.585_{\pm 0.042}$**
QM7	MAE	$118.883_{\pm 2.421}$	$173.582_{\pm 4.293}$	$76.000_{\pm 2.743}$	$213.014_{\pm 10.618}$	$82.681_{\pm 3.979}$	$74.710_{\pm 9.079}$	$108.913_{\pm 25.555}$	$49.661_{\pm 4.246}$	**$39.697_{\pm 3.400}$**

use evaluation metrics from the benchmark paper (Wu et al., 2018), except for the MUV dataset for which we use ROC-AUC following recent studies (Hu et al., 2019; Rong et al., 2020). For the social network tasks, we follow the evaluation metrics from (Xu et al., 2018).

Results. Due to brevity of space, we present the relative performance of AWARE compared to the baselines in Table 1. We present complete results for all tasks with error bounds in Appendix I. In Table 1, we observe that AWARE achieves the best performance for 33 out of the 65 tasks, while being ranked in the top-3 performing methods for 53 tasks. In particular, AWARE (even with a simple fully-connected predictor) significantly outperforms N-Gram Graph (which uses a powerful RF or XGB predictor) in 44 tasks, and achieves comparable performance for all other tasks. This indicates that AWARE can successfully learn a weighting scheme to selectively focus on the graph information that is important for the downstream prediction task. Furthermore, we present complete results for several tasks from different domains in Table 2. We observe that AWARE enjoys strong performance compared to baseline models. Furthermore, being able to weight different walks in the graph, AWARE can improve the performance of N-Gram Graph in these tasks.

Effects of weighting components. We present an ablation study of the impact of the weighting components $\{W_v, W_w, W_g\}$ of AWARE by removing them individually and comparing the performance with the full model. In Table 3, we see that all three weighting components contribute to improved performance for most tasks. Notably, there exist tasks for which specific weights lead to a drop in performance. Aligning with Theorems 1 and 2 in Section 5, this indicates that weighting schemes are *successful* in learning important artifacts for the downstream task only under specific conditions.

7 Interpretation and Visualization

AWARE uses an attention mechanism at the walk level (W_w) to aggregate crucial information from the neighbors of each vertex (Section 4). While we have demonstrated the empirical effectiveness of this in Section 6, we now focus on validating our analysis that AWARE can highlight important substructures of the input graph for the prediction task. For this analysis, we use the MUTAGENICITY dataset (Kazius et al., 2005), which comes with the ground-truth information that the 'mutagen' label is assigned to molecules due to presence of specific chemical groups (-NO_2, -NH_2) (Debnath et al., 1991). To find substructures that AWARE uses for its prediction, we compute the importance score for

Task	No W_v	No W_w	No W_g	No W_v, W_w or W_g
IMDB-BINARY	−5.03%	+1.12%	−1.96%	−7.54%
NR-AR	+1.32%	−0.76%	+0.67%	+0.37%
CT_TOX	−9.00%	−2.09%	+0.70%	−3.07%
FDA_APPROVED	−7.83%	−2.39%	+1.38%	−4.16%
MUV-466	−20.08%	−16.70%	−7.44%	+1.80%
DELANEY	−28.45%	−0.18%	−4.69%	−57.80%
MALARIA	−0.83%	+2.10%	−1.15%	−2.32%
QM7	−11.59%	+3.74%	−18.45%	−71.39%

Table 3: Change in performance on removing weighting components of AWARE. "+" / "-" indicate relative improvement/decrease in performance with respect to the full AWARE model respectively.

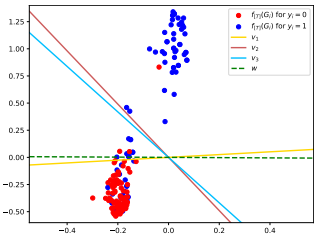
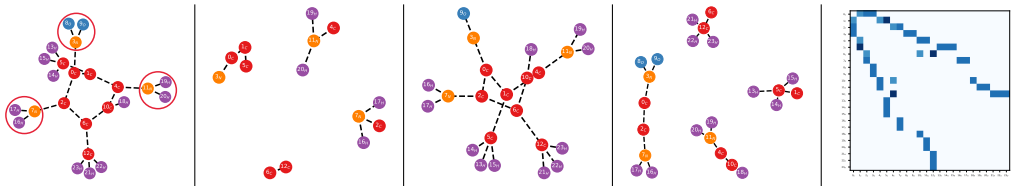


Figure 1: Interpretation of graph-level attention W_g for the NR-AR classification task.



(a) Molecule (b) Grad (c) GNNExplainer (d) AWARE (e) edge importance

Figure 2: Visualization of a random mutagen molecule from MUTAGENICITY and its important substructures for accurate prediction captured by different interpretation techniques. Different node colors indicate different atom types. (a) depicts the original molecule with important mutagenic atom groups circled in red, such as NO_2 and NH_2. (b), (c), and (d) demonstrate important substructures detected by different methods. (e) is a heatmap for the edge importance scores computed by AWARE.

each bond (edge) of the molecule by using the attention scores computed via Eqn (3) (Specifically for an edge $i-j$, we use $[\mathbf{S}_{(T)}]_{ij} + [\mathbf{S}_{(T)}]_{ji}$).

To further elaborate this point, we visualize a randomly chosen 'mutagenic' molecule in Figure 2 and the important substructures as attributed by different interpretation techniques. Figures 2b and 2c depict the interpretation of the GIN model (Xu et al., 2018) using Grad and GNNExplainer techniques (Ying et al., 2019). The former computes gradients with respect to the adjacency matrix and vertex features, while the latter extracts substructures with the closest property prediction to the complete graph. Although both of these techniques are able to highlight the two NH_2 groups as important for the final prediction, they fail to highlight the NO_2 group. For AWARE, we set a threshold ($\geq$1.0) on the importance scores for highlighting important substructures. AWARE can successfully highlight both the NH_2 and NO_2 groups as important in Figure 2d. More examples with interpretation are provided in Appendix G.

Interpretation for W_g. AWARE uses W_g to selectively weight the embeddings at the graph level for the prediction task (Section 4). Towards interpreting W_g, we want to analyze how well it aligns with the predictor for the downstream task. Specifically, we train AWARE for the binary classification NR-AR task (TOX21 dataset) using a linear predictor with parameter w (without a non-linear activation function). We randomly sample 200 data points, and compute their graph embeddings $f_{[T]}(G)$ from AWARE. We denote the top three left singular vectors of W_g by $\{u_1, u_2, u_3\}$. For a particular u_i, we define $v_i = [u_i, u_i, \ldots \text{(T times)}]$ to bring u_i to the same dimensional space as $f_{[T]}(G)$. Finally, we plot $\{v_1, v_2, v_3\}$, w, and the embeddings $f_{[T]}(G)$ for all 200 samples in Figure 1 using PCA with $n = 2$ components. We observe that W_g's largest singular vector direction aligns very well with the parameter w of the downstream predictor. This suggests that this weight can successfully emphasize the directions in the graph embedding space that are important for the prediction.

8 CONCLUSION

In this work, we present and analyze a novel attentive walk-aggregating GNN: AWARE—providing the first provable guarantees on the learning performance of weighted GNNs. Our experiments on 65 tasks from two domains show that AWARE overall outperforms traditional and recent baselines. Our interpretability study lends support to our algorithm design and theoretical insights. Future research directions include the analysis of our algorithm by incorporating additional graph information.

9 ETHICAL STATEMENT

Though AWARE can be used for graph-structured data from distinct data domains, we will highlight the ethical implications of our method for the important domain of molecular property prediction. Having strong empirical performance for the molecular property prediction domain, AWARE can potentially be used for efficient drug development process. Physical experiments for this task can be expensive and slow, which can be alleviated by using AWARE for an initial virtual screening (selecting high-confident candidates from a large pool before physical screening). A strong empirically performing model like AWARE can help speed up the process, and provide tremendous cost savings for this important task. However, deploying an automatic ML prediction model for such a highly critical task must be done extremely carefully. As evidenced by Section 6, AWARE does not achieve the best performance for *all* molecular property prediction tasks. Thus, AWARE may fail to identify promising chemicals for drug development, and/or make erroneous selections. While the former may increase the time and cost of the process, the latter might lead to failures in developing the drug. Nevertheless, with sufficient physical experimentation performed by human experts, such unwanted events can be minimized while still enjoying the benefits of using AWARE.

10 REPRODUCIBILITY STATEMENT

Experiments. To ensure the reproducibility of the empirical results given in Section 6 and Appendix I, we include our code base in the supplementary material, which contains: (1) instructions for installing necessary packages, (2) data preprocessing scripts, (3) training scripts, and (4) optimal hyperparameters to reproduce the results for AWARE and all other baselines. In addition, we provide information about our hyperparameter search and training strategy in Appendix D and Appendix E, respectively. Upon acceptance, we will publicly release our code base to ensure reproducibility of our experiments. Furthermore, full empirical results on 65 tasks along with errors bounds are provided in Appendix I.

Theory. Complete proofs of the theorems and clear explanations of any assumptions made in Section 5 are given in Appendix A.

REFERENCES

AIDS Antiviral Screen Data. Aids antiviral screen data. https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data, 2017. Accessed: 2020-12-20. 27

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020. 1, 2, 7

Sanjeev Arora, Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A compressed sensing view of unsupervised text embeddings, bag-of-n-grams, and lstm. *International Conference on Learning Representations*, 2018. 20, 23, 24

Artem V Artemov, Evgeny Putin, Quentin Vanhaelen, Alexander Aliper, Ivan V Ozerov, and Alex Zhavoronkov. Integrated deep learned transcriptomic and structure-based predictor of clinical trials outcomes. *bioRxiv*, pp. 095653, 2016. 27

Waiss Azizian and marc lelarge. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=lxHgXYN4bwl. 2

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL http://arxiv.org/abs/1409.0473. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. 1

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016. 2

Lorenz C Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database gdb-13. *Journal of the American Chemical Society*, 131(25): 8732–8733, 2009. 27

Robert Calderbank, Sina Jafarpour, and Robert Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. *Techical Report*, 2009. 23

Emmanuel J Candes. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 346(9-10):589–592, 2008. 23

Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005. 23

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. ACM, 2016. 7

Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 518–533, 2018. 2

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33, 2020. 7

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34 (2):786–797, 1991. 8

Nima Dehmamy, Albert-Laszlo Barabasi, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, 2019. 1, 2

John S. Delaney. ESOL: Estimating Aqueous Solubility Directly from Molecular Structure. *Journal of Chemical Information and Computer Sciences*, 44(3):1000–1005, May 2004. ISSN 0095-2338. doi: 10.1021/ci034243x. 27

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423. 1, 3

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 3

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015. 1, 2, 7

Simon Foucart and Holger Rauhut. A mathematical introduction to compressive sensing. *Bull. Am. Math*, 54:151–165, 2017. 23

Francisco-Javier Gamo, Laura M. Sanz, Jaume Vidal, Cristina de Cozar, Emilio Alvarez, Jose-Luis Lavandera, Dana E. Vanderwall, Darren V. S. Green, Vinod Kumar, Samiul Hasan, James R. Brown, Catherine E. Peishoff, Lon R. Cardon, and Jose F. Garcia-Bustos. Thousands of chemical starting points for antimalarial lead identification. *Nature*, 465(7296):305–310, May 2010. ISSN 1476-4687. doi: 10.1038/nature09107. 27

Kaitlyn M Gayvert, Neel S Madhukar, and Olivier Elemento. A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology*, 23(10):1294–1301, 2016. 27

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. 1, 2, 3, 35

Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S. Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M. Brockway, and Alán Aspuru-Guzik. The Harvard Clean Energy Project: Large-Scale Computational Screening and Design of Organic Photovoltaics on the World Community Grid. *The Journal of Physical Chemistry Letters*, 2(17): 2241–2251, September 2011. ISSN 1948-7185. doi: 10.1021/jz200866s. 27

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019. 8

Shiva Prasad Kasiviswanathan and Mark Rudelson. Restricted isometry property under high correlations. *arXiv preprint arXiv:1904.05510*, 2019. 24

Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1):312–320, 2005. 8, 27, 31

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8): 595–608, 2016. 2, 35

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1, 2

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020. 2

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 2

Shengchao Liu, Mehmet F Demirel, and Yingyu Liang. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. In *Advances in Neural Information Processing Systems*, pp. 8466–8478, 2019. 1, 2, 3, 4, 5, 7, 16, 18, 23, 28

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 1

Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1614–1623, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/martins16.html. 3

Lukasz Maziarka, Tomasz Danel, Slawomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanislaw Jastrzkebski. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*, 2020. 2, 3

HL Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965. 7

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019. 2

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014. 1, 2, 3

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html. 3

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 3

Raghunathan Ramakrishnan, Mia Hartmann, Enrico Tapavicza, and O Anatole Von Lilienfeld. Electronic spectra from tddft and machine learning in chemical space. *The Journal of chemical physics*, 143(8):084111, 2015. 27

Sebastian G Rohrer and Knut Baumann. Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. *Journal of chemical information and modeling*, 49 (2):169–184, 2009. 27

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Grover: Self-supervised message passing transformer on large-scale molecular data. *arXiv preprint arXiv:2007.02835*, 2020. 2, 3, 8

Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012. 27

Mark Rudelson, Roman Vershynin, et al. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18, 2013. 25, 26

Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):1–8, 2017. 35

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep): 2539–2561, 2011. 1, 2, 7

Tox21 Data Challenge. Tox21 data challenge 2014. *https://tripod.nih.gov/tox21/challenge/*, 2014. 27

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, 2017. 1

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 2, 3, 7, 28

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015. 3

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *The Journal of Machine Learning Research*, 11:1201–1242, 2010. 1, 2, 3

Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Multi-hop attention graph neural networks. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 3089–3096. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/425. URL https://doi.org/10.24963/ijcai.2021/425. Main Track. 2

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. *Chemical science*, 9(2):513–530, 2018. 8

Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 63(16):8749–8760, 2019. 7

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2048–2057, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/xuc15.html. 1

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 1, 2, 7, 8, 9

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015. 27

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8): 3370–3388, 2019. 2, 28, 35

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 974–983, 2018. 1

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in neural information processing systems*, pp. 9244–9255, 2019. 9

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems*, pp. 11983–11993, 2019. 2

Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10076–10085, 2020. 3

Appendix

An Analysis of Attentive Walk-Aggregating Graph Neural Networks

---

Appendix A: Our complete theoretical analysis, including the proofs for the results in the main text, as well as additional results and discussions.

Appendix B: Details about the datasets that we use in our experiments as well as their license information.

Appendix C: Details about the vertex attribute information.

Appendix D: Details about the yperparameters used in our experiments for AWARE and all other baselines.

Appendix E: Details about our training strategies.

Appendix F: Figures on the effect of certain hyperparameters on model performance.

Appendix G: Further details about the MUTAGENICITY dataset, and additional interpretation examples with statistics of the entire dataset.

Appendix H: Additional information about our main ablation study from Table 3 as well as two extra ablation studies.

Appendix I: Complete experimental results on 65 tasks for AWARE and all baseline methods.

Appendix J: Complete experimental results on 60 molecular property prediction tasks for AWARE and models that use extra edge/3D information.

A COMPLETE THEORETICAL ANALYSIS

In our algorithm, W_v weights the vertex embeddings f_i's, and W_g weights the final embeddings $F_{(n)}$, emphasizing important directions in the corresponding space.[3] On the other hand, the effect of the weighting $\mathbf{S}_{ij}$ imposed by W_w on the messages is unclear. In this section, we provide theoretical analysis for the effect of $\mathbf{S}_{ij}$. We make the following simplification assumption:

- $W_v = W_g = I$, the number of attributes is $C = 1$, and the activation is linear $\sigma(z) = z$.

In this simplified case, the only weighting is $\mathbf{S}_{ij}$, which allows our analysis to focus on its effect. We further assume that the number of attributes on the vertices is $C = 1$ to simplify the notations; the generalization of the analysis to the case with $C > 1$ is straightforward. We also provide analysis for the general case where W_v, W_g are not the identity matrix and $C > 1$ in Section A.3.

We will show that our algorithm incorporates the attention weighting $\mathbf{S}_{ij}$ in a principled way, and thus can potentially highlight important information and reduce irrelevant information for the prediction, which can improve learning. To this end, we first analyze what types of information can be encoded in our graph embedding and how they are weighted, and then analyze why the weighting can help learning a predictor with better performance.

A.1 THE EFFECT OF WEIGHTING ON REPRESENTATION

We will show that the representation/embedding $f_{(n)}$ is a linear mapping of a high dimension vector $c_{(n)}$ into the low dimension embedding space, where the vector $c_{(n)}$ records the statistics about the walks in the graph.

[3] This can be seen by factorizing W_v into its singular vector decomposition $W_v = U\Sigma V^\top$ and observing that W_v will relatively emphasize the singular vector directions with large singular values and depress those with smaller singular values. Similar for W_g.

We first formally define the walk statistics $c_{(n)}$ (a variant of the count statistics defined in (Liu et al., 2019)). Recall that we assume the number of attributes is $C = 1$. K is the number of possible attribute values, and the columns of the vertex embedding parameter matrix $W \in \mathbb{R}^{r \times K}$ are embeddings for different attribute values u. Let $W(u)$ denote the column for value u, i.e., $W(u) = Wh(u)$ where $h(u)$ is the one-hot vector of u.

Definition 3 (Walk Statistics, Restatement of Definition 1)**.** *A walk type of length n is a sequence of n attribute values $v = (v_1, v_2, \cdots, v_n)$. The walk statistics vector $c_{(n)}(G) \in \mathbb{R}^{K^n}$ is the histogram of all walk types of length n in the graph G, i.e., each entry is indexed by a walk type v and the entry value is the number of walks with sequence of attribute values v in the graph. Furthermore, let $c_{[T]}(G)$ is the concatenation of $c_{(1)}(G), \ldots, c_{(T)}(G)$. When G is clear from the context, we write $c_{(n)}$ and $c_{[T]}$ for short.*

We also introduce the following notation for describing the linear mapping projecting $c_{(n)}$ to the representation $f_{(n)}$.

Definition 4 (ℓ-way Column Product)**.** *Let A be a $d \times N$ matrix, and let ℓ be a natural integer. The ℓ-way column product of A is a $d \times N^\ell$ matrix denoted as $A^{[\ell]}$, whose column indexed by a sequence $(i_1, i_2, \cdots, i_\ell)$ is the element-wise product of the $i_1, i_2, \ldots, i_\ell$-th columns of A, i.e., $(i_1, i_2, \ldots, i_\ell)$-th column in $A^{[\ell]}$ is $A_{i_1} \odot A_{i_2} \odot \cdots \odot A_{i_\ell}$ where A_j for $j \in [N]$ is the j-th column in A, and $\odot$ is the element-wise product.*

This definition is for a general matrix A. Then $W^{[n]}$ is an r by K^n matrix, whose columns are indexed by walk types $v = (v_1, v_2, \cdots, v_n)$ and equal $W(v_1) \odot W(v_2) \odot \cdots \odot W(v_n)$. So $W^{[n]}$ matches the definition in the main text.

Definition 5 (Walk Weights)**.** *The weight of a walk type $v = (v_1, \ldots, v_n)$ is*

$$\lambda(v) := \prod_{i=1}^{n-1} S(W(v_i), W(v_{i+1}))$$

where $S(\cdot, \cdot)$ is the weight function in Eq equation 2.

The following theorem then shows that $f_{(n)}$ can be viewed as a compressed version of the walk statistics, weighted by the attention weights $\mathbf{S}$.

Theorem 3 (Restatement of Theorem 1)**.** *The embedding $f_{(n)}$ is a linear mapping of the walk statistics $c_{(n)}$:*

$$f_{(n)} = \mathcal{M}_{(n)} \Lambda_{(n)} c_{(n)}$$

where $\mathcal{M}_{(n)}$ is a matrix depending only on W, and $\Lambda_{(n)}$ is a K^n-dimensional diagonal matrix whose columns are indexed by walk types v and have diagonal entries $\lambda(v)$. Therefore,

$$f_{[T]} := \mathcal{M} \Lambda c_{[T]} \tag{4}$$

where $\mathcal{M}$ is a block-diagonal matrix with diagonal blocks $\mathcal{M}_{(1)}, \mathcal{M}_{(2)}, \ldots, \mathcal{M}_{(T)}$, and Λ is block-diagonal with blocks $\Lambda_{(1)}, \Lambda_{(2)}, \ldots, \Lambda_{(T)}$.

Proof. It is sufficient to prove the first statement with $\mathcal{M}_{(n)} = W^{[n]}$, as the second one directly follows. To this end, we will first prove the following lemma.

Lemma 1. *Let $\mathcal{P}_{i,n}$ be the set of walks starting from vertex i and of length n. Then the latent vector on vertex i is:*

$$[F_{(n)}]_i = \sum_{p \in \mathcal{P}_{i,n}} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right] \tag{5}$$

where $w(v_p)$ is the weight for the sequence of attribute values on p, and $\bigodot_{k \in p} [F_{(1)}]_k$ is the element-wise product of all the $[F_{(1)}]_k$'s on p.

Proof. We prove the lemma by induction. For $n = 1$, it is trivially true.

Suppose the statement is true for $n-1$. Then recall that $[F_{(n)}]_i$ is constructed by weighted-summing up all the latent vectors $[F_{(n-1)}]_j$ from the neighbors j of i, and then element-wise product with $[F_{(1)}]_i = f_i$. So letting $\mathcal{N}_i$ denote the set of neighbors of i, we have by induction

$$[F_{(n)}]_i = \left(\sum_{j \in \mathcal{N}_i} \mathbf{S}_{ji} [F_{(n-1)}]_j \right) \odot [F_{(1)}]_i \tag{6}$$

$$= \sum_{j \in \mathcal{N}_i} \mathbf{S}_{ji} \left(\sum_{p \in \mathcal{P}_{j,n-1}} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right] \right) \odot [F_{(1)}]_i \tag{7}$$

$$= \sum_{j \in \mathcal{N}_i} \sum_{p \in \mathcal{P}_{j,n-1}} \mathbf{S}_{ji} w(v_p) \left([F_{(1)}]_i \odot \left[\bigodot_{k \in p} [F_{(1)}]_k \right] \right). \tag{8}$$

By concatenating i to the walks $p \in \mathcal{P}_{j,n-1}$ for all neighbors $j \in \mathcal{N}_i$, we obtain the set of walks starting from i and of length n, i.e., $\mathcal{P}_{i,n}$. Furthermore, for a path obtained by concatenating i and $p \in \mathcal{P}_{j,n-1}$, the weight is exactly $\mathbf{S}_{ji} \cdot w(v_p)$. Therefore,

$$[F_{(n)}]_i = \sum_{j \in \mathcal{N}_i} \sum_{p \in \mathcal{P}_{j,n-1}} \mathbf{S}_{ji} \cdot w(v_p) \left([F_{(1)}]_i \odot \left[\bigodot_{k \in p} [F_{(1)}]_k \right] \right) \tag{9}$$

$$= \sum_{p \in \mathcal{P}_{i,n}} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right]. \tag{10}$$

By induction, we complete the proof. □

We now use Lemma 1 to prove the theorem statement. Recall that h_k is the one-hot vector for the attribute on vertex k. Let $e_p \in \{0,1\}^{K^n}$ be the one-hot vector for the walk type of a walk p.

$$f_{(n)} = F_{(n)} \mathbf{1} \tag{11}$$

$$= \sum_{i=1}^{m} [F_{(n)}]_i \tag{12}$$

$$= \sum_{i=1}^{m} \sum_{p \in \mathcal{P}_{i,n}} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right] \tag{13}$$

$$= \sum_{p: \text{walks of length } n} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right] \tag{14}$$

$$= \sum_{p: \text{walks of length } n} w(v_p) \left[\bigodot_{k \in p} (W h_k) \right] \tag{15}$$

$$= \sum_{p: \text{walks of length } n} w(v_p) W^{[n]} e_p \tag{16}$$

$$= W^{[n]} \sum_{p: \text{walks of length } n} w(v_p) e_p \tag{17}$$

$$= W^{[n]} \Lambda_{(n)} c_{(n)}. \tag{18}$$

The third line follows from Lemma 1. The forth line follows from that the union of $\mathcal{P}_{i,n}$ for all i is the set of all walks of length n. The sixth line follows from the definition of $W^{[n]}$ and e_p. The last line follows from the definitions of $\Lambda_{(n)}$ and $c_{(n)}$. □

Remark. The theorem shows that the embedding $f_{(n)}$ can encode a compressed version of the weighted walk statistics $\Lambda_{(n)}c_{(n)}$. Note that similar to $\Lambda_{(n)}$, $c_{(n)}$ is in high dimension K^n. Its entries are indexed by all possible sequences of the attribute values $(v_0, \ldots, v_{n-1})$, and the entry value is just the count of the corresponding sequence in the graph. $\Lambda_{(n)}c_{(n)}$ is thus an entry-wise weighted version of the counts, i.e., weighting the walks with attribute $(v_0, \ldots, v_{n-1})$ by $w(v_0, \ldots, v_{n-1})$.

The N-gram graph method is a special case of our method, by setting the message weights $S(\cdot, \cdot)$ to be always 1 (and thus $\Lambda_{(n)}$ being an identity matrix). Then we have $f_{(n)} = W^{[n]}c_{(n)}$. Our method thus enjoys greater representation power, since it can be viewed as a generalization that allows to weight the features. What is more important, and is also the focus of our study, is that this weighting can potentially help learn a predictor with better prediction performance. This is analyzed in the next subsection.

Remark. The weighted walk statistics is then compressed from a high dimension to a low dimension by multiplying with $W^{[n]}$. For the unweighted case, the analysis in (Liu et al., 2019) shows that there exists a large family of W (e.g., the entries of W are independent Rademacher variables) such that $W^{[n]}$ has RIP and thus $c_{(n)}$ can be recovered from $f_{(n)}$ by compressive sensing techniques. A similar result holds for the weighted case.

In particular, it is well known in the compressive sensing literature that when $W^{[n]}$ satisfies the Restricted Isometry Property (RIP), and $\Lambda_{(n)}c_{(n)}$ is sparse, then $\Lambda_{(n)}c_{(n)}$ can be recovered from $f_{(n)}$ (see the review in A.4). That is, $f_{(n)}$ preserve the information of $\Lambda_{(n)}c_{(n)}$. This is indeed the case for a wide family of W.

Theorem 4. *If $r = \Omega((ns_n^3 \log K)/\epsilon^2)$ where s_n is the sparsity of $c_{(n)}$, then there is a prior distribution over W such that with probability $1 - \exp(-\Omega(r^{1/3}))$, $W^{[n]}$ satisfies (s_n, ϵ)-RIP. Therefore, if $r = \Omega(ns_n^3 \log K)$ and $\Lambda_{(n)}c_{(n)}$ is the sparsest vector satisfying $f_{(n)} = W^{[n]}\Lambda_{(n)}c_{(n)}$, then with probability $1 - \exp(-\Omega(r^{1/3}))$, $\Lambda_{(n)}c_{(n)}$ can be recovered from $f_{(n)}$.*

Proof. The first statement follows from Theorem 9 in Appendix A.5, and the second follows from Theorem 7 in Appendix A.4. □

The distribution of W satisfying the above can be that with (properly scaled) i.i.d. Rademacher entries or Gaussian entries. Since this is not the focus of our paper, below we simply assume that W has RIP and focus on analyzing the effect of the weighting on the learning over the representations.

A.2 The Effect of Weighting on Learning

Once we have shown that the embedding $f_{(n)}$ can be viewed as a linear mapping of the weighted walk statistics to low dimensional representations, we are now ready to analyze if the weighting can potentially improve the learning.

The intuition for the benefit of appropriate weighting is simple. To illustrate the intuition, first consider the case where we learn over the weighted features $\Lambda c_{[T]}$ (instead on learning over $f_{[T]}(G) = \mathcal{M}\Lambda c_{[T]}$ which has an additional $\mathcal{M}$). Suppose that the label is given by a linear function on $c_{[T]}$ with parameter θ^*, i.e., $y = \langle \theta^*, c_{[T]} \rangle$. If Λ is invertible, the parameter $\Lambda^{-1}\beta^*$ on $\Lambda c_{[T]}$ has the same loss as β^* on $c_{[T]}$. So we only need to learn $\Lambda^{-1}\beta^*$. The sample size needed to learn $\Lambda^{-1}\beta^*$ on $\Lambda c_{[T]}$ will depend on the factor $\|\Lambda^{-1}\beta^*\|_2 \|\Lambda c_{[T]}\|_2$, which is potentially smaller than $\|\beta^*\|_2 \|c_{[T]}\|_2$ for the unweighted case. This means fewer data samples are needed (equivalently, smaller loss for a fixed amount of samples).

Now, consider the case of learning over $f_{[T]}(G) = \mathcal{M}\Lambda c_{[T]}$ that has an extra $\mathcal{M}$. We note that $c_{[T]}$ can be sparse compared to its high dimension (since likely only a very small fraction of all possible walk types will appear in a graph). Well-established results from compressive sensing show that when $\mathcal{M}$ has the Restricted Isometry Property (RIP), learning over $\mathcal{M}\Lambda c_{[T]}$ is comparable to learning over $\Lambda c_{[T]}$. Indeed, Theorem 4 in Appendix A shows when W is random and the embedding dimension r is large enough, there are families of distributions of W such that $\mathcal{M}$ has RIP for $\Lambda c_{[T]}$. Thus, we assume $\mathcal{M}$ has RIP and focus on the analysis of how W_w affects the weighting and the learning. In practice, our method is more general and the parameters are learned over the data. Still, the analysis

in the special case under the assumptions can provide useful insights for understanding our method, in particular, how the weighting can affect the learning of a predictor over the embeddings.

However, the above intuition is only for learning over a fixed weighting Λ induced by a fixed W_w. Our key challenge is to incorporate the learning of W_w in the analysis. Formally, we consider learning W_w from a hypothesis class $\mathcal{W}$, and let $\Lambda(W_w)$ and $f_{[T]}(G; W_w)$ denote the weights and representation given by W_w. For prediction, we consider binary classification with the logistic loss $\ell(g, y) = \log(1 + \exp(-gy))$ where g is the prediction and y is the true label. Let $\ell_{\mathcal{D}}(\theta, W_w)$ be the risk of a linear classifier with a parameter θ on $f_{[T]}(G; W_w)$ over the data distribution $\mathcal{D}$, and let $\ell_S(\theta, W_w)$ denote the risk over the training dataset S. Suppose we have a dataset $S = \{(G_i, y_i)\}_{i=1}^M$ of M i.i.d. sampled from $\mathcal{D}$, and $\hat{\theta}$ is the parameter over $f_{[T]}(G)$ which is learned via ℓ_2-regularization with regularization coefficient B_θ:

$$\hat{\theta}, \widehat{W_w} = \underset{W_w \in \mathcal{W}, \|\theta\|_2 \le B_\theta}{\arg\min} \ell_S(\theta, W_w) := \frac{1}{M} \sum_{i=1}^{M} \ell\Big(\langle \theta, f_{[T]}(G_i; W_w)\rangle, y_i\Big). \tag{19}$$

To derive error bounds, suppose $\mathcal{W}$ is equipped with a norm $\|\cdot\|$ and let $\mathcal{N}(\mathcal{W}, \epsilon)$ be the ϵ-covering number of $\mathcal{W}$ w.r.t. the norm $\|\cdot\|$ (other complexity measures on $\mathcal{W}$, such as VC-dimension, can also be used). Suppose $f_{[T]}(G; W_w)$ is L_f-Lipschitz w.r.t. the norm $\|\cdot\|$ on $\mathcal{W}$ and the ℓ_2 norm on the representation. Furthermore, let β^* denote the best linear classifier on $c_{[T]}$, and let $\ell^*_{\mathcal{D}}$ denote its risk.

Theorem 5 (Restatement of Theorem 2)**.** *Assume $c_{[T]}$ is s-sparse, $\mathcal{M}$ satisfies $(2s, \epsilon_0)$-RIP, $\Lambda(W_w)$ is invertible and $f_{[T]}(G; W_w)$ is L_f-Lipschitz over $\mathcal{W}$. For any $\delta, \epsilon \in (0, 1)$, there are regularization coefficient values B_θ such that with probability $\ge 1 - \delta$:*

$$\ell_{\mathcal{D}}(\hat{\theta}, \widehat{W_w}) \le \ell^*_{\mathcal{D}} + 2\epsilon + O\left(\sqrt{\frac{rT + \mathcal{C}_\epsilon(\mathcal{W})}{M}}\right) + \min_{W_w \in \mathcal{W}} B(W_w) \times O\left(\sqrt{\epsilon_0 + \frac{\mathcal{C}_\epsilon(\mathcal{W})}{M}}\right)$$

where

$$\mathcal{C}_\epsilon(\mathcal{W}) := \log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right) + \log\frac{1}{\delta}, \quad B(W_w) := \max_{G \sim \mathcal{D}} \|\Lambda(W_w) c_{[T]}(G)\|_2 \|\Lambda(W_w)^{-1} \beta^*\|_2.$$

Proof. Since $\hat{\theta} = \hat{\theta}(\widehat{W})$ where $\hat{\theta}(\widehat{W})$ is defined in Lemma 2, by Lemma 2.(1), we have

$$\ell_{\mathcal{D}}(\hat{\theta}, \widehat{W_w}) \le \ell_S(\hat{\theta}, \widehat{W_w}) + O\left(\sqrt{\frac{1}{M}\left(rT + \log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right) + \log\frac{1}{\delta}\right)}\right) + \epsilon. \tag{20}$$

Furthermore, since $\hat{\theta}, \widehat{W_w}$ are the optimal solution for the regularized regression, then for any $W_w \in \mathcal{W}$,

$$\ell_S(\hat{\theta}, \widehat{W_w}) \le \ell_S(\hat{\theta}(W_w), W_w). \tag{21}$$

Then by Lemma 2.(2), we have

$$\ell_S(\hat{\theta}(W_w), W_w) \le \ell^*_{\mathcal{D}} + O\left(B(W_w)\sqrt{\epsilon_0 + \frac{1}{M}\left(\log\frac{1}{\delta} + \log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right)\right)}\right) + \epsilon. \tag{22}$$

Combining the above inequalities proves the theorem. □

Lemma 2. *Suppose $f_{[T]}(G; W_w)$ is L_f-Lipschitz w.r.t. the norm $\|\cdot\|$ on $\mathcal{W}$ and the ℓ_2 norm on the representation. Let*

$$\hat{\theta}(W_w) = \underset{\|\theta\|_2 \le B_\theta}{\arg\min} \frac{1}{M} \sum_{i=1}^{M} \ell\Big(\langle \theta, f_{[T]}(G_i; W_w)\rangle, y_i\Big) \tag{23}$$

be the optimal solution for a fixed W_w.
(1) For any $\epsilon, \delta \in (0, 1)$, with probability at least $1 - \delta$, for any $W_w \in \mathcal{W}$,

$$|\ell_{\mathcal{D}}(\hat{\theta}(W_w), W_w) - \ell_S(\hat{\theta}(W_w), W_w)| \le O\left(\sqrt{\frac{1}{M}\left(rT + \log \mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right) + \log\frac{1}{\delta}\right)}\right) + \epsilon. \tag{24}$$

(2) Assume that $\mathcal{M}$ satisfies the $(2s, \epsilon_0)$-RIP, and $c_{[T]}$ is s-sparse. Also assume that $\Lambda^{-1}(W_w)$ is invertible over $\mathcal{W}$. Then for any $\epsilon, \delta \in (0,1)$, there exists an appropriate choice of regularization coefficient B_θ, such that with probability at least $1-\delta$, for any $W_w \in \mathcal{W}$,

$$\ell_{\mathcal{D}}(\hat{\theta}(W_w), W_w) \leq \ell^*_{\mathcal{D}} + O\left(B(W_w)\sqrt{\epsilon_0 + \frac{1}{M}\left(\log\frac{1}{\delta} + \log\mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right)\right)}\right) + \epsilon. \quad (25)$$

Proof. (1) We apply a net argument on $\mathcal{W}$. Let $\mathcal{X}$ be an $\epsilon/8B_\theta L_f$-net of $\mathcal{W}$, so $|\mathcal{X}| \leq \mathcal{N}(\mathcal{W}, \epsilon/8B_\theta L_f)$. Then for the given M, any $W_w \in \mathcal{X}$ and any θ satisfies:

$$|\ell_{\mathcal{D}}(\theta, W_w) - \ell_S(\theta, W_w)| \leq O\left(\sqrt{\frac{1}{M}\left(rT + \log\mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right) + \log\frac{1}{\delta}\right)}\right). \quad (26)$$

Then for any $W'_w \in \mathcal{W}$, there exists a $W_w \in \mathcal{X}$ such that $\|W_w - W'_w\| \leq \epsilon/8B_\theta L_f$. Then letting θ denote $\hat{\theta}(W'_w)$, we have

$$\begin{aligned}
|\ell_{\mathcal{D}}(\theta, W'_w) - \ell_S(\theta, W'_w)| &\leq |\ell_{\mathcal{D}}(\theta, W'_w) - \ell_D(\theta, W_w)| && (27)\\
&\quad + |\ell_D(\theta, W_w) - \ell_S(\theta, W_w)| && (28)\\
&\quad + |\ell_S(\theta, W_w) - \ell_S(\theta, W'_w)|. && (29)
\end{aligned}$$

For any G with label y, we have

$$\begin{aligned}
&|\ell(\langle\theta, f_{[T]}(G; W_w)\rangle, y) - \ell(\langle\theta, f_{[T]}(G; W'_w)\rangle, y)| && (30)\\
&\leq |\langle\theta, f_{[T]}(G; W_w)\rangle - \langle\theta, f_{[T]}(G; W'_w)\rangle| && (31)\\
&= |\langle\theta, f_{[T]}(G; W_w) - f_{[T]}(G; W'_w)\rangle| && (32)\\
&= \|\theta\|_2 \|f_{[T]}(G; W_w) - f_{[T]}(G; W'_w)\|_2 && (33)\\
&\leq B_\theta L_f \|W_w - W'_w\| && (34)\\
&\leq \frac{\epsilon}{8}. && (35)
\end{aligned}$$

Then

$$|\ell_{\mathcal{D}}(\theta, W'_w) - \ell_S(\theta, W'_w)| \leq \frac{\epsilon}{8} + O\left(\sqrt{\frac{1}{M}\left(rT + \log\mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right) + \log\frac{1}{\delta}\right)}\right) + \frac{\epsilon}{8}. \quad (36)$$

This proves the first statement.

(2) Let $\mathcal{X}$ be the set of $\Lambda c_{[T]}$ for G from the data distribution. Since $c_{[T]}$ is s-sparse, $\Lambda c_{[T]}$ is also s-sparse. Then $\Lambda c_{[T]}(G) - \Lambda c_{[T]}(G')$ is $2s$-sparse for any G and G', so $\mathcal{M}$ satisfies $(\Delta\mathcal{X}, \epsilon)$-RIP. Then we can apply the theorem for learning over compressive sensing data. In particular, for a fixed W_w, we apply Theorem 4.2 in (Arora et al., 2018). (The theorem is included as Theorem 8 in Section A.4 for completeness. Note that choosing an appropriate λ in that theorem is equivalent to choosing an appropriate B_θ by standard Lagrange multiplier theory.) The statement follows from that the logistic loss function is 1-Lipschitz and convex, and that the optimal solution over $\Lambda(W_w)c_{[T]}$ is $\Lambda^{-1}(W_w)\theta^*$ with the same loss as θ^* over $c_{[T]}$. Combining with a net argument similar as above proves the statement. □

Remark. Theorem 2 shows that the learned model has risk comparable to that of the best linear classifier on the walk statistics, given sufficient data. Let's now compare to the unweighted case (i.e., Λ is the identity matrix), where $\log\mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right)$ reduces to 0, and $B(W_w)$ reduces to

$$B_0 := \max_{G\sim\mathcal{D}} \|c_{[T]}(G)\|_2 \|\beta^*\|_2. \quad (37)$$

So our method needs extra samples to learn W_w, leading to the extra error terms related to $\log\mathcal{N}\left(\mathcal{W}, \frac{\epsilon}{8B_\theta L_f}\right)$. On the other hand, the benefit of weighting is replacing the factor B_0 with

$\min_{W_w} B(W_w)$. If there is W_w^* with $B(W_w^*) \ll B_0$, the error is significantly reduced. Therefore, there is a trade-off between the reduction of error for learning classifiers on an appropriate weighted representation and the additional samples needed for learning an appropriate weighting.

The benefit of weighting can be significant in practice. $\min_{W_w} B(W_w)$ can be much smaller than B_0, especially when some features (i.e., walk types) in $c_{[T]}$ are important while others are not, which is true for many real-world applications.

For a concrete example, suppose $c_{[T]}(G)$ is s-sparse with each non-zero entry being some constant c. Suppose only a few of the features are useful for the prediction, i.e., β^* is ρ-sparse with each non-zero entry being some constant b, and $\rho \ll s$. Suppose there is a weighting W_w^* that leads to weight Υ on the entries corresponding to the ρ important features (i.e., the non-zero entries in β^*), and weight υ for the other features where $|\upsilon| \ll |\Upsilon|$. Then it can be shown that

$$B_0 = \sqrt{sc^2}\sqrt{\rho b^2} = bc\sqrt{\rho s}, \tag{38}$$

$$\min_{W_w} B(W_w) \leq \sqrt{\rho(\Upsilon c)^2 + (s-\rho)(c\upsilon)^2}\sqrt{\rho(b/\Upsilon)^2} \tag{39}$$

and thus

$$\frac{\min_{W_w} B(W_w)}{B_0} \leq \sqrt{\frac{\rho}{s} + \left(1 - \frac{\rho}{s}\right)\left(\frac{\upsilon}{\Upsilon}\right)^2}. \tag{40}$$

Since $\rho \ll s$ and $|\upsilon| \ll |\Upsilon|$, $\min_{W_w} B(W_w)$ is much smaller than B_0, so the weighting can significantly reduce the error. This demonstrate that with proper weighting highlighting important features and depressing irrelevant features for prediction, the error can be much smaller than the error for without weighting.

A.3 Analysis for More General Cases

Here we provide the analysis for the more general case where W_v and W_g are not the identity matrix I and the number of attributes $C > 1$. We still make the assumption that σ is linear, while the analysis for nonlinear σ is left for future work.

We will need to generalize the notations. Given a graph, let us define the walk statistics $c_{(n)}$ as follows (generalizing Definition 1). Recall that C is the number of attributes, k_j is the number of possible values for the j-th attribute. Let $K_C := \prod_{j=1}^{C} k_j$ denote the number of possible attribute value vector. Also, $h_i^j \in \{0,1\}^{k_j}$ is the one hot vector for the j-th attribute on vertex i, and the one hot vector for vertex i is h_i, the concatenation $[h_i^1, \ldots, h_i^C] \in \{0,1\}^K$. The ℓ-th column of the embedding parameter matrix $W^j \in \mathbb{R}^{r \times k_j}$ is an embedding vector for the ℓ-th value of the j-th attribute, and the parameter matrix $W \in \mathbb{R}^{r \times K}$ is the concatenation $W = [W^1, W^1, \ldots, W^C]$ with $K = \sum_{j=0}^{C-1} k_j$. Finally, given an attribute vector $u = [u^1, u^2, \ldots, u^C]$ where u^j is the value for the j-th attribute, let Let $W(u)$ denote the embedding for u, i.e., $W(u) = Wh(u)$ where $h(u) = [h(u^1), h(u^2), \ldots, h(u^C)]$ and $h(u^j)$ is the one-hot vector of u^j.

Definition 6 (Walk Statistics for the General Case)**.** *A walk type of length n is a sequence of n attribute value vectors $\upsilon = (v_1, v_2, \cdots, v_n)$ where v_i's are a vector of C attributes. The walk statistics vector $c_{(n)}(G) \in \mathbb{R}^{K_C^n}$ is the histogram of all walk types of length n in the graph G, i.e., each entry is indexed by a walk type υ and the entry value is the number of walks with sequence of attribute value vectors υ in the graph. Furthermore, let $c_{[T]}(G)$ is the concatenation of $c_{(1)}(G), \ldots, c_{(T)}(G)$. When G is clear from the context, we write $c_{(n)}$ and $c_{[T]}$ for short.*

So the definition is similar to that for the case with $C = 1$, except that now a walk type considers all C attributes. Similarly, the definition of the walk weight is the same as that for $C = 1$, except that the walk type definition is generalized.

To describe the linear mapping from $c_{(n)}$ to $f_{(n)}$, we need to introduce the following notation.

Definition 7. *Let $(W_v W)^{\{n\}}$ be a matrix with K_C^n column corresponding to all possible length-n walk types, with the column indexed by a walk type $\upsilon = (v_1, \ldots, v_n)$ being $(W_v W(v_1)) \odot (W_v W(v_2)) \odot \cdots \odot (W_v W(v_n))$.*

The following theorem then shows that $f_{(n)}$ can be a compressed version of the walk statistics, weighted by the weighting parameter matrix W_v, W_g and also by the attention scores $\mathbf{S}$.

Theorem 6. *The embedding $f_{(n)}$ is a linear mapping of the walk statistics $c_{(n)}$:*

$$f_{(n)} = W_g(W_v W)^{\{n\}} \Lambda_{(n)} c_{(n)}. \tag{41}$$

where $\Lambda_{(n)}$ is a K_C^n-dimensional diagonal matrix, whose columns are indexed by walk types v and have diagonal entries $\lambda(v)$. Therefore,

$$f_{[T]} := \mathcal{M} \Lambda c_{[T]} \tag{42}$$

where $\mathcal{M}$ is a block-diagonal matrix with diagonal blocks $W_g(W_v W), W_g(W_v W)^{\{2\}}, \ldots, W_g(W_v W)^{\{T\}}$, and Λ is block-diagonal with blocks $\Lambda_{(1)}, \Lambda_{(2)}, \ldots, \Lambda_{(T)}$.

Proof. The proof is similar to that of Theorem 1. First, we note that Lemma 1 still applies to the general case. So we can use Lemma 1 to prove the theorem statement. Recall that h_k is the one-hot vector for the attributes on vertex k. Let $e_p \in \{0,1\}^{K_C^n}$ be the one-hot vector for the walk type of a walk p.

$$\begin{aligned}
f_{(n)} &= W_g F_{(n)} \mathbf{1} && (43)\\
&= W_g \sum_{i=1}^{m} [F_{(n)}]_i && (44)\\
&= W_g \sum_{i=1}^{m} \sum_{p \in \mathcal{P}_{i,n}} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right] && (45)\\
&= W_g \sum_{p:\text{walks of length } n} w(v_p) \left[\bigodot_{k \in p} [F_{(1)}]_k \right] && (46)\\
&= W_g \sum_{p:\text{walks of length } n} w(v_p) \left[\bigodot_{k \in p} (W_v W h_k) \right] && (47)\\
&= W_g \sum_{p:\text{walks of length } n} w(v_p) (W_v W)^{\{n\}} e_p && (48)\\
&= W_g (W_v W)^{\{n\}} \sum_{p:\text{walks of length } n} w(v_p) e_p && (49)\\
&= W_g (W_v W)^{\{n\}} \Lambda_{(n)} c_{(n)}. && (50)
\end{aligned}$$

The third line follows from Lemma 1. The forth line follows from that the union of $\mathcal{P}_{i,n}$ for all i is the set of all walks of length n. The sixth line follows from the definitions of $(W_v W)^{\{n\}}$ and e_p. The last line follows from the definition of $\Lambda_{(n)}$ and $c_{(n)}$. □

The theorem shows that in the general case, the embedding $f_{(T)}$ is also a linear mapping of the walk statistics $c_{(T)}$, with a more complicated mapping $W_g(W_v W)^{\{n\}}$. Similarly, with properly set W_g, W_v, W, the linear mapping can satisfy the requirement of compressive sensing, e.g., satisfy RIP. Then $\mathcal{M}$ will also satisfy RIP.

We note that Theorem 2 directly applies to the general case, under the same set of assumptions.

Finally, we observe in our experiments that a nonlinear σ helps the optimization; with a linear σ the training of the network is less stable and leads to worse performance (see the ablation studies in Appendix H). However, the analysis of a nonlinear σ is beyond the scope of this paper. We leave it as an interesting future direction.

A.4 TOOLBOX FROM COMPRESSIVE SENSING

For completeness, here we include the review from (Liu et al., 2019) about related concepts in the field of compressed sensing that are important for our analysis. Please refer to (Foucart & Rauhut, 2017) for more details.

The primary goal of compressed sensing is to recover a high-dimensional k-sparse signal $x \in \mathbb{R}^N$ from a few linear measurements. Here, being k-sparse means that x has at most k non-zero entries, i.e., $|x|_0 \leq k$. In the noiseless case, we have a design matrix $A \in \mathbb{R}^{d \times N}$ and the measurement vector is $z = Ax$. The optimization formulation is then

$$\text{minimize}_{x'} \|x'\|_0 \quad \text{subject to} \quad Ax' = z \tag{51}$$

where $\|x'\|_0$ is ℓ_0 norm of x', i.e., the number of non-zero entries in x'. The assumption that x is the sparsest vector satisfying $Ax = z$ is equivalent to that x is the optimal solution for (51).

Unfortunately, the ℓ_0-minimization in (51) is NP-hard. The typical approach in compressed sensing is to consider its convex surrogate using ℓ_1-minimization:

$$\text{minimize}_{x'} \|x'\|_1 \quad \text{subject to} \quad Ax' = z \tag{52}$$

where $\|x'\|_1 = \sum_i |x'_i|$ is the ℓ_1 norm of x'. The fundamental question is when the optimal solution of (51) is equivalent to that of (52), i.e., when exact recovery is guaranteed.

A.4.1 THE RESTRICTED ISOMETRY PROPERTY

One common condition for recovery is the Restricted Isometry Property (RIP):

Definition 8. *$A \in \mathbb{R}^{d \times N}$ is $(\mathcal{X}, \epsilon)$-RIP for some subset $\mathcal{X} \subseteq \mathbb{R}^N$ if for any $x \in \mathcal{X}$,*

$$(1 - \epsilon)\|x\|_2 \leq \|Ax\|_2 \leq (1 + \epsilon)\|x\|_2.$$

We will abuse notation and say (k, ϵ)-RIP if $\mathcal{X}$ is the set of all k-sparse $x \in \mathbb{R}^N$.

Introduced by (Candes & Tao, 2005), RIP has been used to show to guarantee exact recovery.

Theorem 7 (Restatement of Theorem 1.1 in (Candes, 2008))**.** *Suppose A is $(2k, \epsilon)$-RIP for an $\epsilon < \sqrt{2} - 1$. Let $\hat{x}$ denote the solution to (52), and let x_k denote the vector x with all but the k-largest entries set to zero. Then*

$$\|\hat{x} - x\|_1 \leq C_0 \|x_k - x\|_1$$

and

$$\|\hat{x} - x\|_2 \leq C_0 k^{-1/2} \|x_k - x\|_1.$$

In particular, if x is k-sparse, the recovery is exact.

Furthermore, it has been shown that A is (k, ϵ)-RIP with overwhelming probability when $d = \Omega(k \log \frac{N}{k})$ and $\sqrt{d}A_{ij} \sim \mathcal{N}(0, 1)(\forall i, j)$ or $\sqrt{d}A_{ij} \sim \mathcal{U}\{-1, 1\}(\forall i, j)$. There are also many others types of A with RIP; see (Foucart & Rauhut, 2017).

A.4.2 COMPRESSED LEARNING

Given that Ax preserves the information of sparse x when A is RIP, it is then natural to study the performance of a linear classifier learned on Ax compared to that of the best linear classifier on x. Our analysis will use a theorem from (Arora et al., 2018) that generalizes that of (Calderbank et al., 2009).

Let $\mathcal{X} \subseteq \mathbb{R}^N$ denote

$$\mathcal{X} = \{x : x \in \mathbb{R}^N, \|x\|_0 \leq k, \|x\|_2 \leq B\}.$$

Let $\{(x_i, y_i)\}_{i=1}^M$ be a set of M samples i.i.d. from some distribution over $\mathcal{X} \times \{-1, 1\}$. Let ℓ denote a λ_ℓ-Lipschitz convex loss function. Let $\ell_{\mathcal{D}}(\theta)$ denote the risk of a linear classifier with weight $\theta \in \mathbb{R}^N$, i.e., $\ell_{\mathcal{D}}(\theta) = \mathbb{E}[\ell(\langle \theta, x \rangle, y)]$, and let θ^* denote a minimizer of $\ell_{\mathcal{D}}(\theta)$. Let $\ell^A_{\mathcal{D}}(\theta)$ denote the

risk of a linear classifier with weight $\theta \in \mathbb{R}^d$ over Ax, i.e., $\ell_{\mathcal{D}}^A(\theta_A) = \mathbb{E}[\ell(\langle \theta_A, Ax \rangle, y)]$, and let $\hat{\theta}_A$ denote the weight learned with ℓ_2-regularization over $\{(Ax_i, y_i)\}_i$:

$$\hat{\theta}_A = \arg\min_{\theta} \frac{1}{M} \sum_{i=1}^{M} \ell(\langle \theta, Ax_i \rangle, y_i) + \lambda \|\theta\|_2 \tag{53}$$

where λ is the regularization coefficient.

Theorem 8 (Restatement of Theorem 4.2 in (Arora et al., 2018))**.** *Suppose A is $(\Delta\mathcal{X}, \epsilon)$-RIP. Then with probability at least $1 - \delta$,*

$$\ell_{\mathcal{D}}^A(\hat{\theta}_A) \leq \ell_{\mathcal{D}}(\theta^*) + O\left(\lambda_\ell B \|\theta^*\| \sqrt{\epsilon + \frac{1}{M} \log \frac{1}{\delta}}\right)$$

for appropriate choice of λ. Here, $\Delta\mathcal{X} = \{x - x' : x, x' \in \mathcal{X}\}$ for any $\mathcal{X} \subseteq \mathbb{R}^N$.

A.5 Tools for the Proof of Theorem 4

For the proof, we concern about whether the ℓ-way column product of W has RIP. Existing results in the literature do not directly apply in our case. But following the ideas in Theorem 4.3 in (Kasiviswanathan & Rudelson, 2019), we are able to prove the following theorem for our purpose.

Theorem 9. *Let X be an $n \times d$ matrix, and let R be a $d \times N$ random matrix with independent entries R_{ij} such that $\mathbb{E}[R_{ij}] = 0, \mathbb{E}[R_{ij}^2] = 1$, and $|R_{ij}| \leq \tau$ almost surely. Let $t \geq 2$ be a constant. Let $\epsilon \in (0, 1)$, and let k be an integer satisfying $sr(X) \geq \frac{C\tau^{4t}k^3}{\epsilon^2} \log \frac{N^\ell}{k}$ for some universal constant $C > 0$. Then with probability at least $1 - \exp(-c\epsilon^2 sr(X)/(k^2\tau^{4t}))$ for some universal constant $c > 0$, the matrix $XR^{[t]}/\|X\|_F$ is (k, ϵ)-RIP.*

Here, $\mathrm{sr}(X) = \|X\|_F^2/\|X\|^2$ is the stable rank of X. In our case, we will apply the theorem with X being $\mathbf{I}_{d\times d}/\sqrt{d}$ where $\mathbf{I}_{d\times d} \in \mathbb{R}^{d\times d}$ is the identity matrix.

Proof of Theorem 9. The proof follows the idea in Theorem 4.3 in (Kasiviswanathan & Rudelson, 2019). However, their analysis is for a different type of matrices (ℓ-way Column Hadamard Product). We thus include a proof for our case for completeness.

Let $u \in \mathbb{R}^{d^t}$ be a vector with sparsity k, and its entries indexed by sequences $(i_1, i_2, \ldots, i_t) \in [d]^{\otimes t}$. Let $\ell \in [p]$, and define

$$y_\ell := \sum_{(i_1,i_2,\ldots,i_t)\in[d]^{\otimes t}} u_{(i_1,i_2,\ldots,i_t)} \prod_{j=1}^{t} R_{\ell i_j}. \tag{54}$$

Note that the random variables $y_\ell (\ell \in [p])$ are independent. We will now estimate the ψ_2-norm of y_ℓ and then use the Hanson-Wright inequality (and its corollaries) with a net argument to establish the concentration for the norm of $XR^{[\ell]}u = Xy$ where $y = (y_1, \ldots, y_p)$.

Let $\mathrm{supp}(u)$ be the support of u. By the triangle inequality,

$$\|y_\ell\|_{\psi_2} = \left\| \sum_{(i_1,i_2,\ldots,i_t)\in[d]^{\otimes t}} u_{(i_1,i_2,\ldots,i_t)} \prod_{j=1}^{t} R_{\ell i_j} \right\|_{\psi_2} \tag{55}$$

$$= \sum_{(i_1,i_2,\ldots,i_t)\in\mathrm{supp}(u)} \left\| u_{(i_1,i_2,\ldots,i_t)} \prod_{j=1}^{t} R_{\ell i_j} \right\|_{\psi_2} \tag{56}$$

$$= O\left(\tau^t \|u\|_1\right) \tag{57}$$

$$= O\left(\tau^t \sqrt{k} \|u\|\right). \tag{58}$$

Next, we choose an $(1/2C_2)$-net $\mathcal{N}$ in the set of all k-sparse vectors in C^{d^t-1} such that

$$|\mathcal{N}| \leq \binom{d^t}{k}(6C_2)^k \leq \exp\left(k \log\left(\frac{C_0 d^t}{k}\right)\right). \tag{59}$$

Note that for any k-sparse vector $u \in C^{d^t-1}$, $y = R^{[t]}u = (y_1, \ldots, y_p)$ is a random vector with independent coordinates such that for any $\ell \in [p]$,

$$\mathbb{E}[y_\ell] = 0, \mathbb{E}[y_\ell^2] = \|u\|_2^2, \text{ and } \|y_\ell\|_{\psi_2} \leq C\tau^t\sqrt{k}\|u\|_2. \tag{60}$$

Then by Corollary 1, for any fixed $u \in C^{d^t-1}$ with $|\text{supp}(u)| \leq k$ (and $y = R^{[t]}u$),

$$\Pr\left[|\|Xy\|_2 - \|X\|_F| > \epsilon\|X\|_F\right] \leq 2\exp\left(-\frac{C\epsilon^2}{\max_\ell \|y_\ell\|_{\psi_2}^4}\text{sr}(X)\right) \leq 2\exp\left(-\frac{C_1\epsilon^2}{\tau^{4t}k^2}\text{sr}(X)\right). \tag{61}$$

Together with the union bound over $u \in \mathcal{N}$ and using the assumption on $\text{sr}(X)$, we have

$$\Pr\left[\exists u \in \mathcal{N}, |\|XR^{[t]}u\|_2 - \|X\|_F| > \epsilon\|X\|_F\right] \leq \exp\left(k\log\left(\frac{C_0 d^t}{k}\right)\right) \cdot 2\exp\left(-\frac{C_1\epsilon^2}{\tau^{4t}k^2}\text{sr}(X)\right). \tag{62}$$

Finally, we extend the above argument from the net to all k-sparse vectors. From Corollary 2, we have

$$\Pr\left[\exists I \subseteq [d]^{\otimes t}, |I| = k, \|XR_I^{[t]}\|_2 > C_1\epsilon\|X\|_F\right] \leq \exp\left(-\frac{c_1\epsilon^2}{\tau^{4t}k^2}\text{sr}(X)\right). \tag{63}$$

First assume that the events in equation 62 and equation 63 happen. Any k-sparse vector u can be written as $u = a + b$, where $a \in \mathcal{N}$, and b satisfies $|\text{supp}(b)| \leq k$ and $\|b\|_2 \leq 1/(2C_1)$. Let $I_b = \text{supp}(b) \subseteq [d]^{\otimes t}$ and let $\tilde{b}$ be b restricted to I_b. Let $R_{I_b}^{[t]}$ be the submatrix of $R^{[t]}$ with columns indexed by I_b. Then

$$\begin{aligned}
\|XR^{[t]}u\|_2 &= \|XR^{[t]}a + XR^{[t]}b\|_2 && (64)\\
&\leq \|XR^{[t]}a\|_2 + \|XR^{[t]}b\|_2 && (65)\\
&= \|XR^{[t]}a\|_2 + \|XR_{I_b}^{[t]}\tilde{b}\|_2 && (66)\\
&\leq \|XR^{[t]}a\|_2 + \|XR_{I_b}^{[t]}\|_2\|\tilde{b}\|_2 && (67)\\
&\leq (1+\epsilon)\|X\|_F + \frac{1}{2C_2}\|XR_{I_b}^{[t]}\|_2 && (68)\\
&\leq (1+\epsilon_1)\|X\|_F && (69)
\end{aligned}$$

where the bound on $\|XR^{[t]}a\|_2$ is from equation 62 and the spectrum norm bound for $\|XR_{I_b}^{[t]}\|_2$ is from equation 63. Similarly,

$$\|XR^{[t]}u\|_2 \geq (1-\epsilon_2)\|X\|_F. \tag{70}$$

Adjusting the constants and removing the conditioning completes the proof. □

For proving the above Theorem 9, the Hanson-Wright Inequality and its corollaries are useful. We thus include them here for completeness.

Theorem 10 (Hanson-Wright Inequality (Rudelson et al., 2013)). *Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ be a random vector with independent components x_i which satisfy $\mathbb{E}[x_i] = 0$ and $\|x_i\|_{\psi_2} \leq K$. Let M be an $n \times n$ matrix. Then for every $t \geq 0$,*

$$\Pr\left[\left|x^\top Mx - \mathbb{E}[x^\top Mx]\right| > t\right] \leq 2\exp\left(-c\min\left\{\frac{t^2}{K^4\|M\|_F^2}, \frac{t}{K^2\|M\|_2}\right\}\right). \tag{71}$$

Corollary 1 (Subgaussian Concentration (Rudelson et al., 2013)). *Let M be a fixed $n \times d$ matrix. Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ be a random vector with independent components x_i which satisfies $\mathbb{E}[x_i] = 0, \mathbb{E}[x_i^2] = 1$ and $\|x_i\|_{\psi_2} \leq K$. Then for every $t \geq 0$,*

$$\Pr[|\|Mx\|_2 - \|M\|_F| > t] \leq 2 \exp\left(\frac{-ct^2}{K^4 \|M\|_2^2}\right). \tag{72}$$

Corollary 2 (Spectrum Norm of the Product (Rudelson et al., 2013)). *Let B be a fixed $n \times p$ matrix, and let $G = (G_{ij})$ be a $p \times d$ matrix with independent entries that satisfy: $\mathbb{E}[G_{ij}] = 0, \mathbb{E}[G_{ij}^2] = 1$, and $\|G_{ij}\|_{\psi_2} \leq K$. Then for any $a, b > 0$,*

$$\Pr[\|BG\|_2 > CK^2(a\|B\|_F + b\sqrt{d}\|B\|_2)] \leq 2 \exp\left(-a^2 sr(B) - b^2 d\right). \tag{73}$$

B DATASET DETAILS

In Table S1, we provide details about the benchmark datasets that we use in our experiments. We evaluate AWARE on 65 tasks from 15 datasets in total: 61 from molecular property prediction domain and 4 from social networks. We have a total of 37 classification and 28 regression tasks.

Table S1: Details on the benchmark datasets used in our experiments

Dataset	# of Tasks	Type	Domain
IMDB-BINARY (Yanardag & Vishwanathan, 2015)	1	Classification	Social Network
IMDB-MULTI (Yanardag & Vishwanathan, 2015)	1	Classification	Social Network
REDDIT-BINARY (Yanardag & Vishwanathan, 2015)	1	Classification	Social Network
COLLAB (Yanardag & Vishwanathan, 2015)	1	Classification	Social Network
MUTAGENICITY (Kazius et al., 2005)	1	Classification	Chemistry
TOX21 (Tox21 Data Challenge, 2014)	12	Classification	Chemistry
CLINTOX (Artemov et al., 2016; Gayvert et al., 2016)	2	Classification	Chemistry
HIV (AIDS Antiviral Screen Data, 2017)	1	Classification	Chemistry
MUV (Rohrer & Baumann, 2009)	17	Classification	Chemistry
DELANEY (Delaney, 2004)	1	Regression	Chemistry
MALARIA (Gamo et al., 2010)	1	Regression	Chemistry
CEP (Hachmann et al., 2011)	1	Regression	Chemistry
QM7 (Blum & Reymond, 2009)	1	Regression	Chemistry
QM8 (Ramakrishnan et al., 2015)	12	Regression	Chemistry
QM9 (Ruddigkeit et al., 2012)	12	Regression	Chemistry

Dataset Licenses. The Delaney (Delaney, 2004), CEP (Hachmann et al., 2011), QM7 (Blum & Reymond, 2009), QM9 (Ruddigkeit et al., 2012), MUV (Rohrer & Baumann, 2009), and MUTAGENICITY (Kazius et al., 2005) datasets are all licensed under the Copyright © of the American Chemical Society (ACS) which allows free usage of the data and materials appearing in public domain articles without any permission. The QM8 (Ramakrishnan et al., 2015) dataset is under Creative Commons Attribution (CC BY) license of the American Institute of Physics (AIP) Publishing LLC requiring no permission from the authors and publisher for using publicly released data from the paper. The ClinTox (Gayvert et al., 2016) dataset is under the Copyright © of Elsevier Ltd. which permits usage of public domain works and open access content without author permissions. The Malaria (Gamo et al., 2010) dataset is licensed under Copyright © of Macmillan Publishers Limited that allows usage for personal and noncommercial use. The Tox21 (Tox21 Data Challenge, 2014) dataset was released by NIH National Center for Advancing Translational Sciences for free public usage as a part of a'crowdsourced' data analysis challenge. The HIV (AIDS Antiviral Screen Data, 2017) dataset was released by NIH National Cancer Institute (NCI) for public usage without any confidentiality agreement which allows access to chemical structural data on compounds. The IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, COLLAB (Yanardag & Vishwanathan, 2015) datasets are licensed under ACM Copyright © 2015 under Creative Commons License that allows free usage for non-commercial academic purposes.

C DESCRIPTION OF VERTEX ATTRIBUTES

Molecular Graphs. In general, molecules can be viewed as graphs, where each atom is a vertex with different attributes and each chemical bond corresponds to an edge. Assume that there are m vertices in the graph and denote them as $i \in \{0, 1, ..., m-1\}$. Each vertex m will then possess useful attribute information, such as the atom symbol and whether the atom is acceptor or donor.[4] Such vertex attributes are folded into a vertex attribute matrix $\mathcal{R} \in \{0, 1\}^{m \times C}$ where C is the number of

[4]Note that the vertex attributes are discrete-valued in general. If there are numeric attributes, they can simply be padded to the learned embedding for the other attributes.

attributes on each vertex $i \in \{0, 1, ..., m-1\}$. Here is a concrete example:

$$\mathcal{R}_{i,\cdot} = [\mathcal{R}_{i,0}, \mathcal{R}_{i,1}, \ldots, \mathcal{R}_{i,6}, \mathcal{R}_{i,7}],$$
$$\text{atom symbol } \mathcal{R}_{i,0} \in \{\text{C}, \text{Cl}, \text{I}, \text{F}, \ldots\},$$
$$\text{atom degree } \mathcal{R}_{i,1} \in \{0, 1, 2, 3, 4, 5, 6\},$$
$$\ldots$$
$$\text{is acceptor } \mathcal{R}_{i,6} \in \{0, 1\},$$
$$\text{is donor } \mathcal{R}_{i,7} \in \{0, 1\}.$$

The matrix $\mathcal{R}$ can then be translated into the vertex attribute vector set $\mathcal{V}$ using one-hot vectors for the attributes.

Social Graphs. For the social network graphs that we use in our experiments, we utilize vertex degrees as vertex attributes (i.e., $C = 1$).

D Hyperparameter Tuning

AWARE. We carefully perform a hyperparameter sweeping for AWARE on the different candidate values listed in Table S2.

Table S2: Hyperparameter sweeping for AWARE

Hyperparameters	Candidate values
Learning rate	1e-3, 1e-4
# of linear layers in the predictor: L	1, 2, 3
Maximum walk length: T	3, 6, 9, 12
Vertex embedding dimension: r	100, 300, 500
Random dimension: r'	100, 300, 500
Optimizer	Adam

Baseline Methods. For all the molecular baseline methods other than GAT, D-MPNN, Attentive FP, and PNA, the hyperparameter search strategy outlined in (Liu et al., 2019) has been adopted. For GAT and D-MPNN, we use their reported optimal hyperparameters (Veličković et al., 2017; Yang et al., 2019). For Attentive FP and PNA, we performed a hyperparameter tuning that included their reported optimal hyperparameters. For social network experiments, we perform hyperparameter tuning on PNA and Attentive FP, and use the optimal hyperparameters reported for other baseline methods. In addition, for some of the social network datasets, we only consider graphs whose total number of vertices is less than a certain threshold (REDDIT-BINARY: 200, COLLAB: 100).

E Training Strategies

Training Details. We train AWARE on 9 classification and 6 regression datasets, each of which consisting of multiple tasks, resulting in a total of 37 classification and 28 regression tasks. Each dataset is split into 5 different sets of training, validation, and test sets (i.e., 5 different random seeds) with a respective ratio of 8:1:1. We train the model for 500 epochs and use early stopping on the validation set with a patience of 50 epochs. No learning rate scheduler is used.

GPU Specifications. In general, A NVIDIA GeForce GTX 1080 (8GB) GPU model was used in the training process to obtain the main experimental results. For some of the bigger datasets, we used an NVIDIA A100 (40 GB) GPU model. The code is also submitted as a supplementary material to help with reproducibility.

F Effects of Different Hyperparameters

In this section, we analyze the effect of different hyperparameters on the prediction performance. Appendix F.1 demonstrates the effect of the maximum walk length T and the latent dimension r',

and Appendix F.2 shows the impact of the number of layers L in the final predictor and the vertex embedding dimension r. In general, the performance is quite stable across different hyperparameter values. This indicates that our algorithm is friendly towards hyperparameter tuning.

F.1 Effect of T and r'

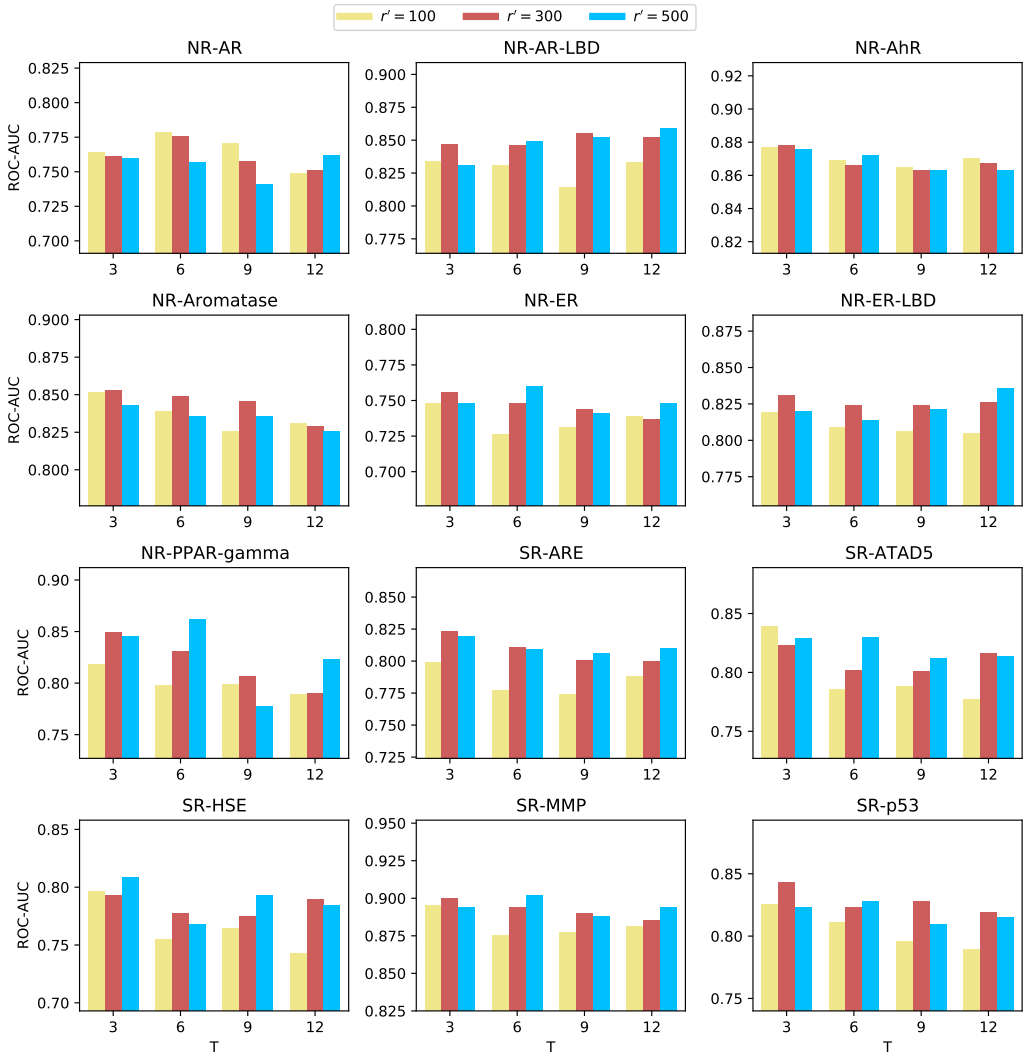


Figure S1: Effect of T and r' on the prediction performance on the 12 tasks in the Tox21 dataset. For each pair of T and r' hyperparameter values, the model was run on 5 different seeds of data and the average of the 5 runs is reported. Higher is better.

F.2 EFFECT OF THE NUMBER OF LAYERS L IN THE PREDICTOR AND VERTEX EMBEDDING DIMENSION r

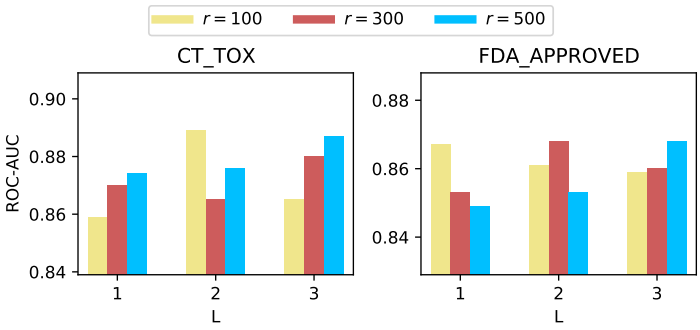


Figure S2: Effect of the number of linear layers L in the fully connected neural network for graph-level prediction and the vertex embedding dimension r on the prediction performance on the 2 tasks in the CLINTOX dataset. For each pair of L and r hyperparameter values, the model was run on 5 different seeds of data and the average of the 5 runs is reported. Higher is better.

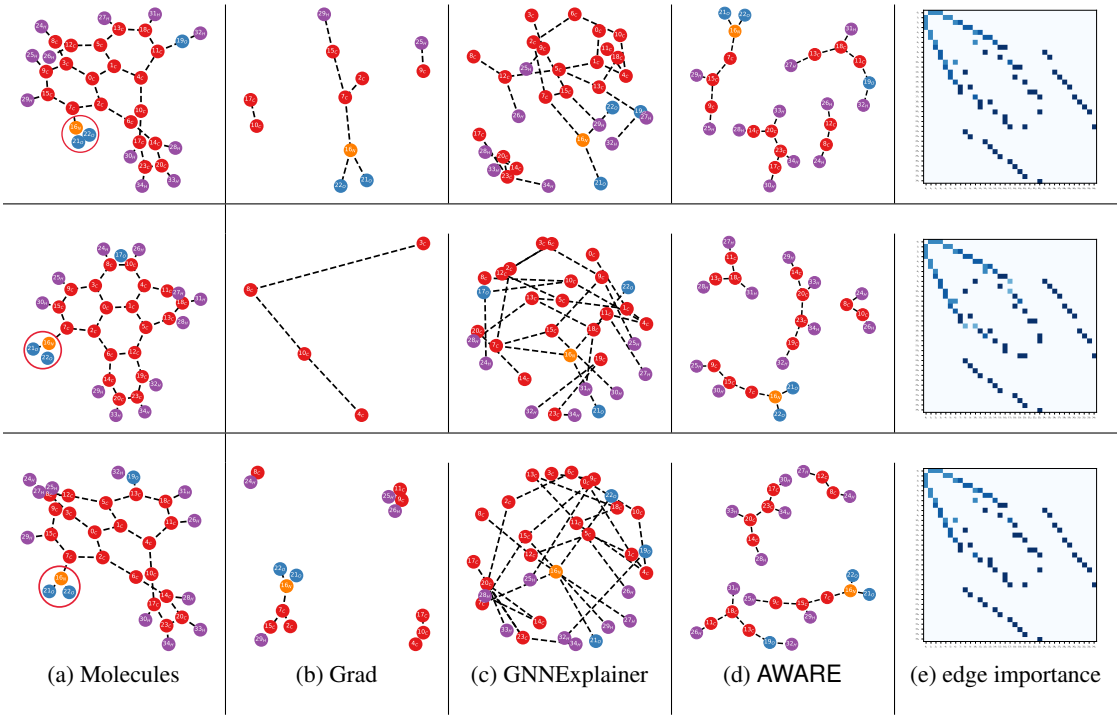
G VISUALIZATION AND INTERPRETATION

G.1 MUTAGENICITY DATASET

The MUTAGENICITY dataset (Kazius et al., 2005) has been introduced for the purpose of increasing accuracy and reliability in mutagenicity predictions for molecular compounds. Mutagenicity of a molecular compound, among many other attributes, is known to impede its ability to become a usable drug. A mutagen is a physical or chemical factor that has the potential to alter the DNA of an organism, which in turn increases the possibility of mutations. The MUTAGENICITY dataset contains 4337 molecular structures with 2401 labeled as "mutagen". Molecular structures in this dataset contain around 30 atoms on average.

G.2 ADDITIONAL EXAMPLES

Here, we present several more examples for the interpretation of our walk attention mechanism in addition to Section 7. It can be seen in Figure S3 that our algorithm successfully finds that the NO_2 and NH_2 atom groups were specifically important for the final model predictions.



(a) Molecules (b) Grad (c) GNNExplainer (d) AWARE (e) edge importance

Figure S3: Additional examples for interpretation from the MUTAGENICITY dataset and their important substructures for accurate prediction captured by different interpretation techniques. Different node colors indicate different atom types. (a) depicts the original molecule with important mutagenic atom groups circled in red, such as NO_2 and NH_2. (b), (c), and (d) demonstrate important substructures detected by different methods. (e) is a heatmap showing the edge importances computed by AWARE ($[\mathbf{S}_{(T)}]_{ij} + [\mathbf{S}_{(T)}]_{ji}$ for edge (i, j)). If the importance is greater than or equal to 1.0, the edge is considered important as in (d). AWARE successfully discovers the important substructures.

H ABLATION STUDIES

Here, we include the results of several ablation study where we try to examine the effects of different variations of our model AWARE. First, we perform an ablation study to examine the impact of each weighting component W_v, W_w and W_g in AWARE. We individually remove one component from the model and compare its performance to the full model. We also compare our full model to the version with linear σ, i.e., $\sigma(z) = z$. Table S3 shows that the weighting components mostly lead to better performance even though there are cases in which they may not. Furthermore, we can also observe the advantage of using a non-linear activation function σ over a linear one. Second, we analyze the change in performance when a non-trainable vertex embedding matrix W and a linear σ are used. Table S4 demonstrates using a trainable random vertex embedding matrix W and a non-linear σ gives overall better performance. It also shows that even with random W and a linear σ, the method can still get decent performance—providing justification for the simplification assumptions in our theoretical analysis. Third, we examine the advantage of using a fully-connected neural network with multiple linear layers as a predictor over using a simple linear predictor. Table S5 suggests that using multiple layers in the final predictor leads to better performance in general.

Table S3: Ablation study I: Change in performance on removing/modifying components of AWARE. "+" / "-" indicate relatively better/worse performance respectively.

Dataset	Task	No W_v	No W_w	No W_g	No W_v, W_w or W_g	Linear σ
IMDB-BINARY	IMDB-BINARY	-5.03%	$+1.12\%$	-1.96%	-7.54%	-10.06%
TOX21	NR-AR	$+1.32\%$	-0.76%	$+0.67\%$	$+0.37\%$	-1.12%
CLINTOX	CT_TOX	-9.00%	-2.09%	$+0.70\%$	-3.07%	-10.35%
CLINTOX	FDA_APPROVED	-7.83%	-2.39%	$+1.38\%$	-4.16%	-10.40%
MUV	MUV-466	-20.08%	-16.70%	-7.44%	$+1.80\%$	-18.73%
DELANEY	DELANEY	-28.45%	-0.18%	-4.69%	-57.80%	-76.17%
MALARIA	MALARIA	-0.83%	$+2.10\%$	-1.15%	-2.32%	-5.86%
QM7	QM7	-11.59%	$+3.74\%$	-18.45%	-71.39%	-85.21%

Table S4: Ablation study II: Change in AWARE's performance when the vertex embedding matrix W is randomly initialized and non-trainable, with linear σ. Underline indicates better performance.

Dataset	Task	Metric	Trainable W	Fixed Random W
IMDB-BINARY	IMDB-BINARY	ACC	0.716	0.660
TOX21	NR-AR	ROC-AUC	0.776	0.774
CLINTOX	CT_TOX	ROC-AUC	0.889	0.764
CLINTOX	FDA_APPROVED	ROC-AUC	0.869	0.774
DELANEY	DELANEY	RMSE	0.612	1.162
MALARIA	MALARIA	RMSE	1.062	1.126
QM7	QM7	MAE	41.280	96.675

Table S5: Ablation study III: Change in AWARE's performance when the final predictor is changed from a multiple layer NN to a linear predictor. Underline indicates better performance.

Dataset	Task	Metric	Multiple layers	Linear predictor
IMDB-BINARY	IMDB-BINARY	ACC	0.716	0.678
TOX21	NR-AR	ROC-AUC	0.776	0.759
CLINTOX	CT_TOX	ROC-AUC	0.889	0.880
CLINTOX	FDA_APPROVED	ROC-AUC	0.869	0.870
DELANEY	DELANEY	RMSE	0.612	0.640
MALARIA	MALARIA	RMSE	1.062	1.070
QM7	QM7	MAE	41.280	415.155

I FULL RESULTS ON ALL CLASSIFICATION AND REGRESSION TASKS

Here, we present complete experimental results on 61 molecular property predictions tasks as well as 4 social network tasks with standard deviations.Table S6 shows the results on the 4 social network classification tasks. Table S7 and Table S8 present the results on the 33 classification and 28 regression tasks in the molecular property prediction domain, respectively.

Table S6: In this table, we present the performance of 8 models on 4 classification tasks in the domain of social networks (Morgan FP is excluded as it works on molecular graphs). Experiments are run on 5 different random seeds, and the average of the 5 reported for each task along with their standard deviation in the subscript. The top-3 models in each task are highlighted in gray and the best one is highlighted in **blue**. Higher is better.

Task	# of Classes	Metric	WL Kernel	GCNN	GAT	GIN	Attentive FP	PNA	N-Gram Graph	AWARE
IMDB-BINARY	2	ACC	$0.680_{\pm 0.022}$	$0.698_{\pm 0.026}$	$0.568_{\pm 0.047}$	$0.696_{\pm 0.037}$	$0.716_{\pm 0.022}$	$0.710_{\pm 0.011}$	$0.522_{\pm 0.036}$	**$0.740_{\pm 0.020}$**
IMDB-MULTI	3	ACC	$0.403_{\pm 0.027}$	$0.459_{\pm 0.033}$	$0.366_{\pm 0.025}$	$0.473_{\pm 0.031}$	$0.481_{\pm 0.021}$	$0.489_{\pm 0.031}$	$0.341_{\pm 0.019}$	**$0.499_{\pm 0.026}$**
REDDIT-BINARY	2	ACC	$0.892_{\pm 0.017}$	$0.931_{\pm 0.013}$	$0.900_{\pm 0.036}$	$0.933_{\pm 0.009}$	$0.864_{\pm 0.029}$	$0.938_{\pm 0.010}$	$0.764_{\pm 0.026}$	**$0.949_{\pm 0.014}$**
COLLAB	3	ACC	$0.567_{\pm 0.011}$	$0.660_{\pm 0.009}$	$0.616_{\pm 0.029}$	$0.669_{\pm 0.014}$	$0.653_{\pm 0.012}$	$0.675_{\pm 0.024}$	$0.376_{\pm 0.119}$	**$0.739_{\pm 0.017}$**

Table S7: In this table, we present the performance of 9 models on 33 classification tasks from the domain of molecular property prediction. Experiments are run on 5 different random seeds, and the average of the 5 reported for each task along with their standard deviation in the subscript. The top-3 models in each task are highlighted in gray and the best one is highlighted in **blue**. We mark incompatible task/model pairs with a "–". Higher is better.

Task	Metric	Morgan FP	WL Kernel	GCNN	GAT	GIN	Attentive FP	PNA	N-Gram Graph	AWARE
MUTAGENICITY	ACC	–	$0.684_{\pm 0.083}$	**$0.758_{\pm 0.011}$**	$0.601_{\pm 0.017}$	$0.747_{\pm 0.019}$	$0.657_{\pm 0.029}$	$0.753_{\pm 0.013}$	$0.506_{\pm 0.011}$	$0.757_{\pm 0.040}$
NR-AR	ROC	$0.763_{\pm 0.043}$	$0.701_{\pm 0.068}$	$0.762_{\pm 0.035}$	$0.754_{\pm 0.058}$	$0.759_{\pm 0.048}$	$0.783_{\pm 0.035}$	$0.786_{\pm 0.039}$	$0.776_{\pm 0.049}$	**$0.786_{\pm 0.041}$**
NR-AR-LBD	ROC	$0.858_{\pm 0.048}$	$0.861_{\pm 0.053}$	$0.844_{\pm 0.046}$	$0.800_{\pm 0.056}$	$0.830_{\pm 0.046}$	$0.839_{\pm 0.065}$	$0.838_{\pm 0.045}$	**$0.873_{\pm 0.039}$**	$0.865_{\pm 0.054}$
NR-AHR	ROC	$0.890_{\pm 0.010}$	$0.876_{\pm 0.017}$	$0.886_{\pm 0.017}$	$0.823_{\pm 0.020}$	$0.872_{\pm 0.016}$	$0.878_{\pm 0.011}$	**$0.901_{\pm 0.013}$**	$0.897_{\pm 0.008}$	$0.889_{\pm 0.006}$
NR-AROMATASE	ROC	$0.821_{\pm 0.024}$	$0.818_{\pm 0.027}$	$0.828_{\pm 0.024}$	$0.744_{\pm 0.039}$	$0.760_{\pm 0.053}$	$0.844_{\pm 0.019}$	$0.837_{\pm 0.018}$	$0.852_{\pm 0.013}$	**$0.861_{\pm 0.019}$**
NR-ER	ROC	$0.726_{\pm 0.036}$	$0.704_{\pm 0.031}$	$0.737_{\pm 0.018}$	$0.706_{\pm 0.042}$	$0.683_{\pm 0.021}$	$0.747_{\pm 0.014}$	$0.738_{\pm 0.030}$	$0.754_{\pm 0.020}$	**$0.765_{\pm 0.028}$**
NR-ER-LBD	ROC	$0.838_{\pm 0.043}$	$0.799_{\pm 0.033}$	$0.813_{\pm 0.048}$	$0.764_{\pm 0.023}$	$0.772_{\pm 0.032}$	$0.808_{\pm 0.037}$	$0.815_{\pm 0.039}$	$0.834_{\pm 0.030}$	**$0.853_{\pm 0.059}$**
NR-PPAR-GAMMA	ROC	$0.840_{\pm 0.063}$	$0.845_{\pm 0.060}$	$0.816_{\pm 0.036}$	$0.758_{\pm 0.035}$	$0.780_{\pm 0.062}$	$0.848_{\pm 0.053}$	$0.841_{\pm 0.067}$	$0.857_{\pm 0.053}$	**$0.862_{\pm 0.040}$**
SR-ARE	ROC	$0.820_{\pm 0.016}$	$0.801_{\pm 0.029}$	$0.809_{\pm 0.014}$	$0.735_{\pm 0.020}$	$0.794_{\pm 0.020}$	$0.809_{\pm 0.028}$	$0.821_{\pm 0.019}$	**$0.851_{\pm 0.014}$**	$0.828_{\pm 0.011}$
SR-ATAD5	ROC	$0.850_{\pm 0.017}$	$0.814_{\pm 0.020}$	$0.827_{\pm 0.052}$	$0.754_{\pm 0.052}$	$0.803_{\pm 0.050}$	$0.807_{\pm 0.047}$	$0.821_{\pm 0.055}$	**$0.853_{\pm 0.025}$**	$0.841_{\pm 0.025}$
SR-HSE	ROC	$0.797_{\pm 0.019}$	$0.803_{\pm 0.037}$	$0.774_{\pm 0.037}$	$0.686_{\pm 0.038}$	$0.740_{\pm 0.062}$	$0.787_{\pm 0.037}$	$0.778_{\pm 0.027}$	$0.808_{\pm 0.025}$	**$0.820_{\pm 0.026}$**
SR-MMP	ROC	$0.890_{\pm 0.007}$	$0.875_{\pm 0.017}$	$0.877_{\pm 0.017}$	$0.834_{\pm 0.014}$	$0.872_{\pm 0.025}$	$0.895_{\pm 0.018}$	$0.873_{\pm 0.019}$	$0.905_{\pm 0.015}$	**$0.905_{\pm 0.014}$**
SR-P53	ROC	$0.844_{\pm 0.012}$	$0.842_{\pm 0.044}$	$0.818_{\pm 0.015}$	$0.733_{\pm 0.036}$	$0.817_{\pm 0.026}$	$0.804_{\pm 0.026}$	$0.843_{\pm 0.024}$	**$0.860_{\pm 0.019}$**	$0.852_{\pm 0.030}$
CT_TOX	ROC	$0.813_{\pm 0.036}$	$0.830_{\pm 0.057}$	$0.860_{\pm 0.027}$	$0.828_{\pm 0.075}$	$0.859_{\pm 0.063}$	$0.873_{\pm 0.053}$	$0.895_{\pm 0.043}$	$0.849_{\pm 0.024}$	**$0.905_{\pm 0.038}$**
FDA_APPROVED	ROC	$0.795_{\pm 0.084}$	$0.862_{\pm 0.029}$	$0.866_{\pm 0.028}$	**$0.899_{\pm 0.033}$**	$0.883_{\pm 0.025}$	$0.870_{\pm 0.070}$	$0.879_{\pm 0.022}$	$0.852_{\pm 0.044}$	$0.895_{\pm 0.050}$
HIV	ROC	**$0.856_{\pm 0.012}$**	$0.811_{\pm 0.015}$	$0.813_{\pm 0.014}$	$0.783_{\pm 0.015}$	$0.829_{\pm 0.014}$	$0.796_{\pm 0.016}$	$0.822_{\pm 0.013}$	$0.843_{\pm 0.017}$	$0.825_{\pm 0.014}$
MUV-466	ROC	$0.765_{\pm 0.142}$	$0.708_{\pm 0.130}$	$0.736_{\pm 0.061}$	$0.749_{\pm 0.109}$	$0.705_{\pm 0.134}$	$0.574_{\pm 0.161}$	$0.713_{\pm 0.085}$	$0.724_{\pm 0.100}$	**$0.830_{\pm 0.078}$**
MUV-548	ROC	$0.953_{\pm 0.036}$	$0.917_{\pm 0.061}$	$0.960_{\pm 0.022}$	$0.764_{\pm 0.117}$	$0.793_{\pm 0.113}$	$0.865_{\pm 0.056}$	$0.966_{\pm 0.016}$	$0.925_{\pm 0.061}$	**$0.976_{\pm 0.016}$**
MUV-600	ROC	$0.536_{\pm 0.098}$	$0.536_{\pm 0.106}$	$0.570_{\pm 0.091}$	$0.437_{\pm 0.095}$	$0.575_{\pm 0.153}$	$0.508_{\pm 0.128}$	$0.680_{\pm 0.111}$	$0.675_{\pm 0.108}$	**$0.687_{\pm 0.062}$**
MUV-644	ROC	$0.893_{\pm 0.068}$	**$0.944_{\pm 0.028}$**	$0.885_{\pm 0.024}$	$0.762_{\pm 0.161}$	$0.749_{\pm 0.094}$	$0.776_{\pm 0.133}$	$0.913_{\pm 0.069}$	$0.799_{\pm 0.085}$	$0.909_{\pm 0.029}$
MUV-652	ROC	$0.725_{\pm 0.131}$	$0.653_{\pm 0.139}$	$0.694_{\pm 0.177}$	$0.493_{\pm 0.124}$	$0.645_{\pm 0.071}$	$0.593_{\pm 0.111}$	$0.659_{\pm 0.124}$	$0.688_{\pm 0.117}$	**$0.819_{\pm 0.084}$**
MUV-689	ROC	$0.676_{\pm 0.277}$	$0.735_{\pm 0.217}$	$0.671_{\pm 0.257}$	$0.553_{\pm 0.247}$	$0.775_{\pm 0.088}$	$0.452_{\pm 0.220}$	$0.666_{\pm 0.172}$	$0.669_{\pm 0.203}$	**$0.833_{\pm 0.077}$**
MUV-692	ROC	**$0.693_{\pm 0.199}$**	$0.447_{\pm 0.193}$	$0.581_{\pm 0.235}$	$0.626_{\pm 0.170}$	$0.629_{\pm 0.118}$	$0.581_{\pm 0.174}$	$0.618_{\pm 0.209}$	$0.606_{\pm 0.147}$	$0.639_{\pm 0.194}$
MUV-712	ROC	$0.927_{\pm 0.058}$	$0.889_{\pm 0.072}$	$0.936_{\pm 0.038}$	$0.760_{\pm 0.162}$	$0.773_{\pm 0.195}$	**$0.946_{\pm 0.040}$**	$0.881_{\pm 0.119}$	$0.812_{\pm 0.103}$	$0.931_{\pm 0.059}$
MUV-713	ROC	$0.554_{\pm 0.206}$	**$0.787_{\pm 0.093}$**	$0.731_{\pm 0.109}$	$0.586_{\pm 0.109}$	$0.567_{\pm 0.183}$	$0.526_{\pm 0.094}$	$0.648_{\pm 0.093}$	$0.715_{\pm 0.089}$	$0.781_{\pm 0.151}$
MUV-733	ROC	$0.709_{\pm 0.101}$	$0.707_{\pm 0.108}$	$0.751_{\pm 0.129}$	$0.637_{\pm 0.053}$	$0.558_{\pm 0.198}$	$0.664_{\pm 0.136}$	$0.632_{\pm 0.168}$	$0.696_{\pm 0.084}$	**$0.819_{\pm 0.127}$**
MUV-737	ROC	$0.791_{\pm 0.092}$	$0.773_{\pm 0.071}$	$0.796_{\pm 0.082}$	$0.675_{\pm 0.087}$	$0.723_{\pm 0.093}$	$0.794_{\pm 0.063}$	$0.810_{\pm 0.111}$	$0.879_{\pm 0.049}$	**$0.917_{\pm 0.058}$**
MUV-810	ROC	$0.794_{\pm 0.111}$	**$0.875_{\pm 0.052}$**	$0.714_{\pm 0.124}$	$0.588_{\pm 0.166}$	$0.682_{\pm 0.188}$	$0.604_{\pm 0.084}$	$0.782_{\pm 0.133}$	$0.680_{\pm 0.094}$	$0.820_{\pm 0.103}$
MUV-832	ROC	**$0.986_{\pm 0.014}$**	$0.964_{\pm 0.034}$	$0.926_{\pm 0.042}$	$0.923_{\pm 0.036}$	$0.918_{\pm 0.129}$	$0.714_{\pm 0.121}$	$0.960_{\pm 0.037}$	$0.969_{\pm 0.030}$	$0.973_{\pm 0.027}$
MUV-846	ROC	$0.877_{\pm 0.128}$	$0.884_{\pm 0.066}$	$0.911_{\pm 0.067}$	$0.863_{\pm 0.151}$	$0.764_{\pm 0.112}$	$0.857_{\pm 0.094}$	$0.940_{\pm 0.024}$	$0.781_{\pm 0.100}$	**$0.964_{\pm 0.027}$**
MUV-852	ROC	$0.890_{\pm 0.096}$	$0.867_{\pm 0.109}$	$0.882_{\pm 0.099}$	$0.743_{\pm 0.133}$	$0.735_{\pm 0.194}$	$0.863_{\pm 0.047}$	$0.850_{\pm 0.086}$	$0.834_{\pm 0.141}$	**$0.917_{\pm 0.090}$**
MUV-858	ROC	$0.701_{\pm 0.080}$	$0.677_{\pm 0.186}$	$0.705_{\pm 0.106}$	$0.650_{\pm 0.205}$	$0.746_{\pm 0.134}$	$0.553_{\pm 0.147}$	**$0.760_{\pm 0.110}$**	$0.630_{\pm 0.148}$	$0.657_{\pm 0.186}$
MUV-859	ROC	$0.530_{\pm 0.082}$	$0.533_{\pm 0.094}$	$0.613_{\pm 0.173}$	$0.499_{\pm 0.076}$	$0.607_{\pm 0.126}$	$0.681_{\pm 0.095}$	$0.604_{\pm 0.037}$	**$0.724_{\pm 0.145}$**	$0.653_{\pm 0.186}$

Table S8: In this table, we present the performance of 9 models on 28 regression tasks from the domain of molecular property prediction. Experiments are run on 5 different random seeds, and the average of the 5 reported for each task along with their standard deviation in the subscript. The top-3 models in each task are highlighted in gray and the best one is highlighted in **blue**. Models that are too slow are left blank. Lower is better.

Task	Metric	Morgan FP	WL Kernel	GCNN	GAT	GIN	Attentive FP	PNA	N-Gram Graph	AWARE
DELANEY	RMSE	$1.081_{\pm 0.073}$	$1.160_{\pm 0.050}$	$0.762_{\pm 0.151}$	$0.954_{\pm 0.151}$	$0.840_{\pm 0.070}$	$0.615_{\pm 0.026}$	$0.922_{\pm 0.122}$	$0.744_{\pm 0.068}$	$\mathbf{0.585_{\pm 0.042}}$
MALARIA	RMSE	$\mathbf{0.995_{\pm 0.028}}$	$1.090_{\pm 0.037}$	$1.141_{\pm 0.057}$	$1.136_{\pm 0.035}$	$1.129_{\pm 0.032}$	$1.080_{\pm 0.028}$	$1.048_{\pm 0.022}$	$1.030_{\pm 0.039}$	$1.056_{\pm 0.036}$
CEP	RMSE	$1.274_{\pm 0.047}$	$1.783_{\pm 0.083}$	$1.457_{\pm 0.112}$	$1.344_{\pm 0.112}$	$\mathbf{1.064_{\pm 0.057}}$	$1.108_{\pm 0.046}$	$1.153_{\pm 0.052}$	$1.409_{\pm 0.029}$	$1.233_{\pm 0.040}$
QM7	MAE	$118.883_{\pm 2.421}$	$173.582_{\pm 4.293}$	$76.000_{\pm 2.743}$	$213.014_{\pm 10.618}$	$82.681_{\pm 3.979}$	$74.710_{\pm 9.079}$	$108.913_{\pm 25.555}$	$49.661_{\pm 4.246}$	$\mathbf{39.697_{\pm 3.400}}$
E1-CC2	MAE	$0.009_{\pm 0.000}$	$0.033_{\pm 0.001}$	$0.007_{\pm 0.001}$	$0.012_{\pm 0.002}$	$0.008_{\pm 0.001}$	$0.012_{\pm 0.001}$	$0.008_{\pm 0.001}$	$0.007_{\pm 0.000}$	$\mathbf{0.007_{\pm 0.000}}$
E2-CC2	MAE	$0.011_{\pm 0.000}$	$0.024_{\pm 0.001}$	$\mathbf{0.007_{\pm 0.000}}$	$0.012_{\pm 0.001}$	$0.008_{\pm 0.000}$	$0.013_{\pm 0.001}$	$0.010_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.008_{\pm 0.000}$
F1-CC2	MAE	$0.016_{\pm 0.001}$	$0.071_{\pm 0.001}$	$0.016_{\pm 0.002}$	$0.020_{\pm 0.003}$	$0.014_{\pm 0.001}$	$0.020_{\pm 0.002}$	$0.015_{\pm 0.001}$	$0.015_{\pm 0.000}$	$\mathbf{0.013_{\pm 0.000}}$
F2-CC2	MAE	$0.035_{\pm 0.001}$	$0.080_{\pm 0.001}$	$0.033_{\pm 0.001}$	$0.038_{\pm 0.001}$	$0.031_{\pm 0.001}$	$0.039_{\pm 0.001}$	$0.032_{\pm 0.001}$	$0.030_{\pm 0.001}$	$\mathbf{0.030_{\pm 0.002}}$
E1-PBE0	MAE	$0.009_{\pm 0.000}$	$0.034_{\pm 0.001}$	$\mathbf{0.006_{\pm 0.001}}$	$0.015_{\pm 0.004}$	$0.007_{\pm 0.001}$	$0.012_{\pm 0.000}$	$0.008_{\pm 0.001}$	$0.007_{\pm 0.000}$	$0.007_{\pm 0.000}$
E2-PBE0	MAE	$0.011_{\pm 0.000}$	$0.029_{\pm 0.001}$	$\mathbf{0.007_{\pm 0.000}}$	$0.012_{\pm 0.002}$	$0.008_{\pm 0.000}$	$0.012_{\pm 0.001}$	$0.009_{\pm 0.001}$	$0.007_{\pm 0.000}$	$0.008_{\pm 0.000}$
F1-PBE0	MAE	$0.014_{\pm 0.000}$	$0.067_{\pm 0.001}$	$0.012_{\pm 0.000}$	$0.016_{\pm 0.001}$	$\mathbf{0.011_{\pm 0.001}}$	$0.017_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.012_{\pm 0.000}$	$0.011_{\pm 0.001}$
F2-PBE0	MAE	$0.028_{\pm 0.001}$	$0.078_{\pm 0.000}$	$0.025_{\pm 0.001}$	$0.030_{\pm 0.001}$	$0.024_{\pm 0.001}$	$0.031_{\pm 0.001}$	$0.025_{\pm 0.000}$	$0.024_{\pm 0.000}$	$\mathbf{0.022_{\pm 0.001}}$
E1-CAM	MAE	$0.009_{\pm 0.000}$	$0.033_{\pm 0.001}$	$\mathbf{0.006_{\pm 0.001}}$	$0.012_{\pm 0.003}$	$0.007_{\pm 0.001}$	$0.012_{\pm 0.001}$	$0.007_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.006_{\pm 0.000}$
E2-CAM	MAE	$0.010_{\pm 0.000}$	$0.026_{\pm 0.001}$	$\mathbf{0.006_{\pm 0.000}}$	$0.011_{\pm 0.001}$	$0.007_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.009_{\pm 0.000}$	$0.007_{\pm 0.000}$	$0.007_{\pm 0.000}$
F1-CAM	MAE	$0.015_{\pm 0.001}$	$0.072_{\pm 0.001}$	$0.013_{\pm 0.000}$	$0.018_{\pm 0.001}$	$0.012_{\pm 0.001}$	$0.017_{\pm 0.001}$	$0.013_{\pm 0.001}$	$0.013_{\pm 0.001}$	$\mathbf{0.012_{\pm 0.001}}$
F2-CAM	MAE	$0.030_{\pm 0.001}$	$0.080_{\pm 0.001}$	$0.027_{\pm 0.001}$	$0.034_{\pm 0.003}$	$0.027_{\pm 0.001}$	$0.035_{\pm 0.003}$	$0.027_{\pm 0.001}$	$0.026_{\pm 0.001}$	$\mathbf{0.024_{\pm 0.001}}$
MU	MAE	$0.625_{\pm 0.003}$	–	$0.506_{\pm 0.019}$	$0.654_{\pm 0.011}$	$\mathbf{0.476_{\pm 0.008}}$	$0.562_{\pm 0.020}$	$0.575_{\pm 0.012}$	$0.536_{\pm 0.002}$	$0.535_{\pm 0.007}$
ALPHA	MAE	$3.348_{\pm 0.018}$	–	$\mathbf{0.533_{\pm 0.083}}$	$1.033_{\pm 0.144}$	$0.688_{\pm 0.081}$	$1.076_{\pm 0.157}$	$3.322_{\pm 0.661}$	$0.595_{\pm 0.004}$	$0.774_{\pm 0.035}$
HOMO	MAE	$0.007_{\pm 0.000}$	–	$0.004_{\pm 0.000}$	$0.008_{\pm 0.001}$	$\mathbf{0.004_{\pm 0.000}}$	$0.009_{\pm 0.000}$	$0.007_{\pm 0.001}$	$0.005_{\pm 0.000}$	$0.006_{\pm 0.000}$
LUMO	MAE	$0.009_{\pm 0.000}$	–	$0.004_{\pm 0.000}$	$0.009_{\pm 0.002}$	$\mathbf{0.004_{\pm 0.000}}$	$0.009_{\pm 0.000}$	$0.008_{\pm 0.001}$	$0.005_{\pm 0.001}$	$0.005_{\pm 0.000}$
GAP	MAE	$0.010_{\pm 0.000}$	–	$0.006_{\pm 0.000}$	$0.011_{\pm 0.001}$	$\mathbf{0.005_{\pm 0.000}}$	$0.012_{\pm 0.000}$	$0.010_{\pm 0.001}$	$0.007_{\pm 0.000}$	$0.007_{\pm 0.000}$
R2	MAE	$97.768_{\pm 0.405}$	–	$\mathbf{30.788_{\pm 2.295}}$	$100.926_{\pm 8.128}$	$36.583_{\pm 1.937}$	$82.265_{\pm 8.864}$	$97.403_{\pm 18.507}$	$56.776_{\pm 0.283}$	$83.000_{\pm 8.780}$
ZPVE	MAE	$0.008_{\pm 0.000}$	–	$0.001_{\pm 0.000}$	$0.004_{\pm 0.002}$	$0.001_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.008_{\pm 0.001}$	$\mathbf{0.000_{\pm 0.000}}$	$0.001_{\pm 0.000}$
CV	MAE	$1.422_{\pm 0.010}$	–	$\mathbf{0.229_{\pm 0.014}}$	$0.541_{\pm 0.220}$	$0.248_{\pm 0.013}$	$0.521_{\pm 0.062}$	$1.318_{\pm 0.256}$	$0.334_{\pm 0.004}$	$0.586_{\pm 0.042}$
U0	MAE	$14.657_{\pm 0.153}$	–	$0.906_{\pm 0.337}$	$1.698_{\pm 1.589}$	$2.283_{\pm 0.567}$	$2.715_{\pm 1.299}$	$22.330_{\pm 3.091}$	$0.427_{\pm 0.032}$	$\mathbf{0.090_{\pm 0.017}}$
U298	MAE	$14.647_{\pm 0.148}$	–	$1.126_{\pm 0.494}$	$5.110_{\pm 5.487}$	$2.032_{\pm 0.453}$	$2.683_{\pm 1.263}$	$21.365_{\pm 2.566}$	$0.428_{\pm 0.032}$	$\mathbf{0.086_{\pm 0.009}}$
H298	MAE	$14.650_{\pm 0.146}$	–	$0.785_{\pm 0.292}$	$2.066_{\pm 1.159}$	$2.308_{\pm 0.580}$	$2.930_{\pm 1.093}$	$20.880_{\pm 5.738}$	$0.429_{\pm 0.032}$	$\mathbf{0.098_{\pm 0.007}}$
G298	MAE	$14.651_{\pm 0.149}$	–	$0.646_{\pm 0.169}$	$2.576_{\pm 1.555}$	$2.269_{\pm 0.596}$	$4.014_{\pm 1.422}$	$19.794_{\pm 3.679}$	$0.427_{\pm 0.028}$	$\mathbf{0.086_{\pm 0.010}}$

J ADDITIONAL EXPERIMENTS WITH MODELS THAT USE EXTRA EDGE/3D INFORMATION

Although using extra edge/3D information is not in the standard setting (as defined in Section 3), in this section, we include the complete additional results comparing AWARE to several models that use extra edge/3D information in their representation learning process (on 60 molecular property prediction tasks, excluding MUTAGENICITY). Even though we are aware that there exist more recent extra edge/3D approaches, we compare AWARE to several important ones: Weave Neural Network (Kearnes et al., 2016) and Directed Message Passing Neural Networks (D-MPNN) (Yang et al., 2019) which use extra edge information, and Deep Tensor Neural Networks (DTNN) (Schütt et al., 2017) and Message Passing Neural Networks (MPNN) (Gilmer et al., 2017) which use extra 3D position information of the vertices. Tables S9 and S10 exhibit the results on the classification and regression tasks, respectively. The experimental results demonstrate that AWARE remains competitive even without any extra edge/3D information, outperforming the others in 39 tasks. This indicates that AWARE exploits the vertex attributes and neighborhood information more effectively.

Table S9: Performance of models that use extra edge/3D information on classification tasks. The experiments are run on 5 different random seeds, and the average of the 5 reported for each task along with their standard deviation in the subscript. The best model in each task is highlighted in **blue**. ROC-AUC is used as the evaluation metric. Higher is better.

Task	Weave	D-MPNN	AWARE
NR-AR	$0.774_{\pm 0.011}$	$0.729_{\pm 0.057}$	$\mathbf{0.786_{\pm 0.041}}$
NR-AR-LBD	$0.824_{\pm 0.052}$	$0.816_{\pm 0.079}$	$\mathbf{0.865_{\pm 0.069}}$
NR-AHR	$0.857_{\pm 0.020}$	$\mathbf{0.892_{\pm 0.013}}$	$0.884_{\pm 0.010}$
NR-AROMATASE	$0.827_{\pm 0.026}$	$0.818_{\pm 0.039}$	$\mathbf{0.861_{\pm 0.019}}$
NR-ER	$0.736_{\pm 0.019}$	$0.724_{\pm 0.035}$	$\mathbf{0.760_{\pm 0.017}}$
NR-ER-LBD	$0.809_{\pm 0.048}$	$0.813_{\pm 0.040}$	$\mathbf{0.851_{\pm 0.056}}$
NR-PPAR-GAMMA	$0.803_{\pm 0.029}$	$0.813_{\pm 0.046}$	$\mathbf{0.862_{\pm 0.040}}$
SR-ARE	$0.771_{\pm 0.032}$	$\mathbf{0.833_{\pm 0.013}}$	$0.827_{\pm 0.030}$
SR-ATAD5	$0.765_{\pm 0.042}$	$0.819_{\pm 0.050}$	$\mathbf{0.841_{\pm 0.023}}$
SR-HSE	$0.749_{\pm 0.043}$	$0.753_{\pm 0.034}$	$\mathbf{0.810_{\pm 0.023}}$
SR-MMP	$0.886_{\pm 0.021}$	$0.891_{\pm 0.018}$	$\mathbf{0.905_{\pm 0.017}}$
SR-P53	$0.787_{\pm 0.033}$	$0.834_{\pm 0.019}$	$\mathbf{0.841_{\pm 0.020}}$
CT_TOX	$0.844_{\pm 0.031}$	$0.841_{\pm 0.068}$	$\mathbf{0.905_{\pm 0.038}}$
FDA_APPROVED	$0.822_{\pm 0.074}$	$0.881_{\pm 0.021}$	$\mathbf{0.895_{\pm 0.050}}$
HIV	$0.556_{\pm 0.059}$	$\mathbf{0.831_{\pm 0.015}}$	$0.823_{\pm 0.008}$
MUV-466	$0.634_{\pm 0.148}$	$0.776_{\pm 0.044}$	$\mathbf{0.830_{\pm 0.078}}$
MUV-548	$0.821_{\pm 0.071}$	$0.932_{\pm 0.037}$	$\mathbf{0.973_{\pm 0.011}}$
MUV-600	$0.575_{\pm 0.148}$	$0.559_{\pm 0.145}$	$\mathbf{0.687_{\pm 0.062}}$
MUV-644	$0.786_{\pm 0.089}$	$0.806_{\pm 0.143}$	$\mathbf{0.883_{\pm 0.060}}$
MUV-652	$0.721_{\pm 0.133}$	$0.539_{\pm 0.155}$	$\mathbf{0.813_{\pm 0.107}}$
MUV-689	$0.576_{\pm 0.300}$	$0.620_{\pm 0.274}$	$\mathbf{0.833_{\pm 0.077}}$
MUV-692	$0.545_{\pm 0.166}$	$\mathbf{0.654_{\pm 0.133}}$	$0.574_{\pm 0.146}$
MUV-712	$0.854_{\pm 0.065}$	$0.797_{\pm 0.111}$	$\mathbf{0.920_{\pm 0.050}}$
MUV-713	$0.686_{\pm 0.116}$	$0.593_{\pm 0.040}$	$\mathbf{0.781_{\pm 0.151}}$
MUV-733	$\mathbf{0.819_{\pm 0.101}}$	$0.746_{\pm 0.153}$	$0.819_{\pm 0.127}$
MUV-737	$0.784_{\pm 0.033}$	$0.805_{\pm 0.079}$	$\mathbf{0.879_{\pm 0.077}}$
MUV-810	$0.593_{\pm 0.165}$	$0.665_{\pm 0.123}$	$\mathbf{0.795_{\pm 0.120}}$
MUV-832	$0.844_{\pm 0.128}$	$0.946_{\pm 0.056}$	$\mathbf{0.969_{\pm 0.021}}$
MUV-846	$0.892_{\pm 0.058}$	$0.870_{\pm 0.131}$	$\mathbf{0.964_{\pm 0.027}}$
MUV-852	$0.859_{\pm 0.060}$	$0.785_{\pm 0.119}$	$\mathbf{0.917_{\pm 0.090}}$
MUV-858	$0.655_{\pm 0.155}$	$\mathbf{0.749_{\pm 0.142}}$	$0.657_{\pm 0.186}$
MUV-859	$0.609_{\pm 0.150}$	$0.465_{\pm 0.193}$	$\mathbf{0.653_{\pm 0.186}}$

Table S10: Performance of models that use extra edge/3D information on regression tasks. The experiments are run on 5 different random seeds, and the average of the 5 reported for each task along with their standard deviation in the subscript. Since only QM8 and QM9 tasks include 3D information, DTNN and MPNN are evaluated only on those. The top-3 models in each task are highlighted in gray and the best one is highlighted in **blue**. RMSE is used as the evaluation metric for the first three tasks and MAE for the rest. Lower is better.

Task	Weave	DTNN	MPNN	D-MPNN	AWARE
DELANEY	$0.620_{\pm 0.121}$	–	–	$0.696_{\pm 0.065}$	$\mathbf{0.589_{\pm 0.051}}$
MALARIA	$1.471_{\pm 0.071}$	–	–	$1.070_{\pm 0.025}$	$\mathbf{1.061_{\pm 0.036}}$
CEP	$2.573_{\pm 0.292}$	–	–	$\mathbf{1.116_{\pm 0.048}}$	$1.252_{\pm 0.023}$
QM7	$59.690_{\pm 3.075}$	–	–	$68.938_{\pm 5.893}$	$\mathbf{40.304_{\pm 3.670}}$
E1-CC2	$0.007_{\pm 0.001}$	$0.007_{\pm 0.000}$	$\mathbf{0.006_{\pm 0.000}}$	$0.006_{\pm 0.000}$	$0.007_{\pm 0.000}$
E2-CC2	$0.008_{\pm 0.000}$	$0.007_{\pm 0.000}$	$\mathbf{0.007_{\pm 0.000}}$	$0.007_{\pm 0.001}$	$0.008_{\pm 0.000}$
F1-CC2	$0.018_{\pm 0.001}$	$0.022_{\pm 0.001}$	$0.020_{\pm 0.001}$	$0.014_{\pm 0.001}$	$\mathbf{0.013_{\pm 0.000}}$
F2-CC2	$0.035_{\pm 0.001}$	$0.041_{\pm 0.000}$	$0.039_{\pm 0.001}$	$0.031_{\pm 0.001}$	$\mathbf{0.030_{\pm 0.002}}$
E1-PBE0	$0.007_{\pm 0.001}$	$\mathbf{0.006_{\pm 0.001}}$	$0.007_{\pm 0.002}$	$0.007_{\pm 0.001}$	$0.007_{\pm 0.000}$
E2-PBE0	$0.007_{\pm 0.000}$	$0.007_{\pm 0.000}$	$\mathbf{0.006_{\pm 0.000}}$	$0.008_{\pm 0.001}$	$0.008_{\pm 0.000}$
F1-PBE0	$0.015_{\pm 0.002}$	$0.018_{\pm 0.001}$	$0.018_{\pm 0.002}$	$0.012_{\pm 0.001}$	$\mathbf{0.011_{\pm 0.000}}$
F2-PBE0	$0.027_{\pm 0.002}$	$0.034_{\pm 0.001}$	$0.032_{\pm 0.002}$	$0.024_{\pm 0.001}$	$\mathbf{0.023_{\pm 0.000}}$
E1-CAM	$0.006_{\pm 0.000}$	$0.006_{\pm 0.000}$	$\mathbf{0.006_{\pm 0.000}}$	$0.006_{\pm 0.001}$	$0.006_{\pm 0.000}$
E2-CAM	$0.006_{\pm 0.000}$	$0.007_{\pm 0.000}$	$0.006_{\pm 0.000}$	$\mathbf{0.006_{\pm 0.000}}$	$0.007_{\pm 0.000}$
F1-CAM	$0.016_{\pm 0.001}$	$0.019_{\pm 0.001}$	$0.019_{\pm 0.001}$	$0.012_{\pm 0.001}$	$\mathbf{0.012_{\pm 0.001}}$
F2-CAM	$0.030_{\pm 0.002}$	$0.036_{\pm 0.001}$	$0.034_{\pm 0.002}$	$0.026_{\pm 0.000}$	$\mathbf{0.024_{\pm 0.001}}$
MU	$0.617_{\pm 0.013}$	$\mathbf{0.244_{\pm 0.006}}$	$0.283_{\pm 0.072}$	$0.464_{\pm 0.013}$	$0.535_{\pm 0.007}$
ALPHA	$0.680_{\pm 0.076}$	$\mathbf{0.387_{\pm 0.043}}$	$0.696_{\pm 0.094}$	$0.408_{\pm 0.026}$	$0.774_{\pm 0.035}$
HOMO	$0.005_{\pm 0.000}$	$\mathbf{0.003_{\pm 0.000}}$	$0.004_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.006_{\pm 0.000}$
LUMO	$0.005_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.004_{\pm 0.000}$	$\mathbf{0.003_{\pm 0.000}}$	$0.005_{\pm 0.000}$
GAP	$0.007_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.006_{\pm 0.000}$	$\mathbf{0.005_{\pm 0.000}}$	$0.007_{\pm 0.000}$
R2	$33.679_{\pm 2.587}$	$\mathbf{10.485_{\pm 2.982}}$	$26.759_{\pm 14.689}$	$41.083_{\pm 4.077}$	$83.000_{\pm 8.780}$
ZPVE	$0.001_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.002_{\pm 0.001}$	$\mathbf{0.000_{\pm 0.000}}$	$0.001_{\pm 0.000}$
CV	$0.254_{\pm 0.025}$	$\mathbf{0.125_{\pm 0.032}}$	$0.280_{\pm 0.092}$	$0.173_{\pm 0.006}$	$0.586_{\pm 0.042}$
U0	$2.419_{\pm 1.339}$	$1.429_{\pm 0.558}$	$3.478_{\pm 3.140}$	$0.673_{\pm 1.024}$	$\mathbf{0.090_{\pm 0.017}}$
U298	$1.375_{\pm 0.245}$	$0.880_{\pm 0.344}$	$2.665_{\pm 3.479}$	$0.557_{\pm 0.791}$	$\mathbf{0.086_{\pm 0.009}}$
H298	$1.472_{\pm 0.221}$	$0.959_{\pm 0.424}$	$1.189_{\pm 0.421}$	$0.522_{\pm 0.708}$	$\mathbf{0.098_{\pm 0.007}}$
G298	$1.948_{\pm 0.691}$	$0.809_{\pm 0.356}$	$1.264_{\pm 0.413}$	$0.581_{\pm 0.555}$	$\mathbf{0.086_{\pm 0.010}}$