

# Re-visiting Skip-Gram Negative Sampling: Dimension Regularization for More Efficient Dissimilarity Preservation in Graph Embeddings

David Liu  
Northeastern University  
Boston, USA  
liu.davi@northeastern.edu

Tina Eliassi-Rad  
Northeastern University  
Boston, USA  
t.eliasirad@northeastern.edu

Arjun Seshadri  
Amazon  
San Francisco, USA  
aseshadr@stanford.edu

Johan Ugander  
Stanford University  
Stanford, USA  
jugander@stanford.edu

## ABSTRACT

A wide range of graph embedding objectives decompose into two components: one that attracts the embeddings of nodes that are perceived as similar, and another that repels embeddings of nodes that are perceived as dissimilar. Because real-world graphs are sparse and the number of dissimilar pairs grows quadratically with the number of nodes, Skip-Gram Negative Sampling (SGNS) has emerged as a popular and efficient repulsion approach. SGNS repels each node from a sample of dissimilar nodes, as opposed to all dissimilar nodes. In this work, we show that node-wise repulsion is, in aggregate, an approximate re-centering of the node embedding dimensions. Such dimension operations are much more scalable than node operations. The dimension approach, in addition to being more efficient, yields a simpler geometric interpretation of the repulsion. Our result extends findings from the self-supervised learning literature to the skip-gram model, establishing a connection between skip-gram node contrast and dimension regularization. We show that in the limit of large graphs, under mild regularity conditions, the original node repulsion objective converges to optimization with dimension regularization. We use this observation to propose an algorithm augmentation framework that speeds up any existing algorithm, supervised or unsupervised, using SGNS. The framework prioritizes node attraction and replaces SGNS with dimension regularization. We instantiate this generic framework for LINE and node2vec and show that the augmented algorithms preserve downstream performance while dramatically increasing efficiency.

## 1 INTRODUCTION

Graph embedding algorithms use the structure of graphs to learn node-level embeddings. Across unsupervised and supervised graph embedding algorithms, their loss functions serve the two roles of preserving *similarity* and *dissimilarity*. Nodes that are similar in the input graph should have similar embeddings, while dissimilar nodes should have dissimilar embeddings [4]. The push and pull of the similarity and dissimilarity objectives are key: in the absence of a dissimilarity objective, the loss would be minimized by embedding all nodes at a single embedding point, a degenerate and useless embedding. Often, enforcing dissimilarity is much more expensive than similarity, owing to the generally sparse nature

of graphs and the number of pairs of dissimilar nodes growing quadratically with the size of the graph. Enforcing dissimilarity is also complex for graphs because graph data frequently have missing edges or noise [18, 31]. In this paper, we show that while many past works have focused on explicitly repelling pairs of dissimilar nodes, the repulsion can be replaced with a regularization of the embedding dimensions. The dimension-based approach improves both scalability and representation quality.

The skip-gram (SG) model is one of the most popular approaches to graph embeddings [1, 28] and can be decomposed into preserving similarity and dissimilarity. Further, skip-gram negative sampling (SGNS) is the dominant method to efficiently approximate dissimilarity preservation. Instead of repelling all pairs of dissimilar nodes, SGNS repels only a sample of dissimilar nodes per pair of similar nodes. SGNS is utilized in LINE [24] and node2vec [9], for instance, and has been shown to yield strong downstream performance in practice. However, several analytical issues with SGNS have also been identified. First, SGNS approximates the original loss function but also introduces a bias by re-scaling the relative importance of preserving similarity and dissimilarity [21]. Second, with SGNS, in the limit as the number of nodes in the graph approaches infinity, the similarities among embeddings diverge from the similarities among nodes in the graph [7]. Although SGNS has been used to learn both graph and word embeddings [16, 17], we focus on the graph context because, for graph embeddings in particular, SGNS remains a popular method for preserving dissimilarity [5].

In this paper, we propose a change in perspective and show that node repulsion in the SG model can be achieved via dimension centering. We draw inspiration from recent advances in the self-supervised learning (SSL) literature, which show an equivalence between sample-contrastive learning and dimension-contrastive learning [2, 8]. Sample-contrastive learning explicitly repels dissimilar pairs while dimension-contrastive learning repels the dimensions from each other. The findings from the SSL literature allow us to directly re-interpret the spectral graph embedding loss function, for example, as achieving dissimilarity by imposing orthogonality among the embedding dimensions. In the case of spectral embeddings, repelling all pairs of nodes is exactly equal to repelling all pairs of dimensions, which is much more scalable given that the number of dimensions is much less than the number of nodes.

The known parallels between sample and dimension contrast, however, do not suggest whether SG loss functions can be also re-interpreted from the dimension perspective. In this paper, we begin by characterizing the degenerate embedding behavior when the dissimilarity term is removed altogether. We prove that, under mild initialization conditions, when only positive edges are considered, the embeddings collapse into a lower dimensional space, which also commonly occurs in self-supervised learning [12]. However, as the dimensions approach collapse, the dissimilarity term also approaches a dimension regularizer that centers the embeddings around the origin. Our findings show that while the dissimilarity term in the SG loss is not itself a dimension regularizer, when the term is most needed to counteract the similarity attraction, dissimilarity preservation can be achieved via regularization.

We operationalize the dimension-based replacement with an algorithm augmentation. We augment existing algorithms using SGNS by making two modifications. First, the augmentation prioritizes the similarity-preservation component, disregarding dissimilarity when possible. This is also desirable because, in real-world graph data, the lack of similarity between two nodes does not necessarily suggest the two nodes are dissimilar; it is also possible that data are missing or noisy [18, 31]. Second, when the embeddings begin to collapse after optimizing only for similarity preservation for a fixed number of epochs, our augmentation repels nodes from each other using a dimension regularizer that aims to re-center the embeddings about the origin.

In summary, our contributions are as follows:

- (1) In Section 2, we introduce a framework that maps node repulsion to dimension regularization. We show that instead of shortcutting the full skip-gram loss function with SGNS and repelling a sample of pairs, the negative function can be approximated with a dimension regularization inducing origin-centered embeddings. We prove that as the need for node repulsion grows, optimizing the regularizer converges to optimizing the skip-gram loss. This framework extends the equivalence between sample-contrastive objectives and dimension regularization established in self-supervised learning.
- (2) In Section 3 we introduce a generic algorithm augmentation that replaces SGNS with dimension regularization for any existing skip-gram embedding algorithm. We instantiate the augmentation for node2vec and LINE, reducing the repulsion complexity from  $\mathcal{O}(n)$  to  $\mathcal{O}(d)$  per epoch.
- (3) In Section 4 we present empirical evaluations of our augmented LINE and node2vec algorithms. Our results show that replacing SGNS with dimension regularization greatly reduces runtime while also improving performance on transductive link prediction. We also show that dimension regularization especially outperforms baselines when graph connectivity is high in a synthetic example, when the blocks in a stochastic block model (SBM) are less distinguishable.

## 2 FROM NODE REPULSION TO DIMENSION REGULARIZATION

In this section, we introduce our loss decomposition framework where a function  $P$  operationalizes similarity preservation and a

**Table 1: Notations used in this paper.**

Symbol	Meaning
$G, V, E$	Graph $G$ with vertices $V$ and edges $E$
$n, m$	number of nodes and edges respectively
$d$	number of embedding dimensions
$N, P$	negative and positive loss functions
$S$	similarity matrix $\in \mathbb{R}^{n \times n}$
$X$	node embedding matrix $\in \mathbb{R}^{n \times d}$
$X_i$	$i^{\text{th}}$ row of $X$ , as a column vector
$X_{.j}$	$j^{\text{th}}$ column of $X$ , as a column vector
$P_\alpha$	Probability distribution with parameter $\alpha$
$k$	number of negative samples per positive update
$\eta$	learning rate
$D_x$	diagonal matrix where $x$ is the diagonal
$C$	the constriction of the embeddings (Def. 2.1)
$\bar{1}, \mathbf{1}$	a vector and matrix of all ones, respectively.

“negative” function  $N$  achieves dissimilarity preservation. We then show that instead of optimizing negative functions with costly node repulsions, we can instead optimize via dimension regularization. Crucially, we show in Subsection 2.2 we show that when node repulsion is needed, the negative function in the skip-gram loss can be optimized via dimension regularization.

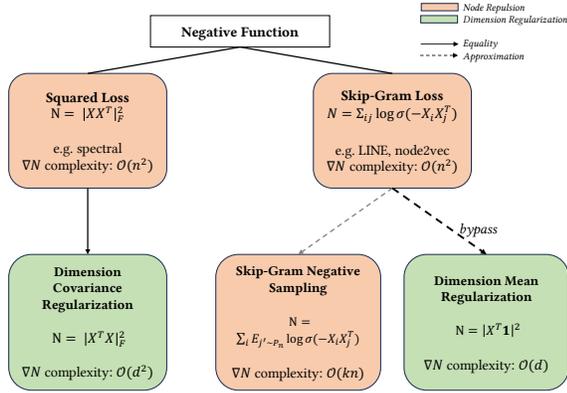
Using notation introduced in Table 1, the decomposition is as follows: given an embedding matrix  $X \in \mathbb{R}^{n \times d}$  and a similarity matrix  $S \in \{0, 1\}^{n \times n}$ , where  $S_{ij} = 1$  if nodes  $i$  and  $j$  are similar, a generic graph embedding loss function  $L(X, S)$  can be written as:

$$L(X, S) = P(X, S) + N(X, S). \quad (1)$$

The decomposition in Equation (1) applies to nearly all unsupervised graph embedding objectives as well as many supervised learning objectives, where supervision is provided in the form of node labels. In the recent graph embedding survey by Chami et al. [5], the decomposition applies to all unsupervised methods except for Graph Factorization [1], which does not include a negative function  $N$ . Examples of popular decomposable loss functions are matrix reconstruction error (e.g., spectral embeddings) as well as softmax (e.g. node2vec [9] and LINE [24]). The decomposition also applies to supervised methods that regularize for graph structure ( $\beta > 0$  as defined in Chami et al. [5]), such as Neural Graph Machines [3] and Planetoid [27].

In the context of graphs, performing gradient descent on the negative example loss  $N$  is a costly operation as the gradient repels all pairs of nodes, resulting in  $\mathcal{O}(n^2)$  time complexity. In this paper, we build on the argument that the costly *node-wise* operation can be replaced with a more efficient *dimension-wise* operation. Optimizing from the dimension perspective also yields a geometric interpretation. This interpretation is illustrated in Figure 1.

Below, we will map the two dominant negative functions found in graph embedding algorithms to dimension regularizations. The two loss functions are spectral loss functions and skip-gram loss functions. The mapping of spectral loss functions to dimension covariance regularization directly ports over a result from the self-supervised learning literature on non-contrastive learning [2, 8]. Our novel contribution is a mapping from the skip-gram loss to a



**Figure 1: Nearly all unsupervised and many supervised graph embedding loss functions define a “negative” function that repels embeddings of dissimilar nodes. We show that instead of repelling pairs of nodes (orange), which is costly, the negative function in the popular skip-gram (SG) loss can be approximated with a dimension regularizer. The regularizer penalizes non-zero dimension means and is efficient given that  $d \ll n$ . This result complements the duality between squared-loss and dimension covariance established in the self-supervised learning literature.**

regularizer that induces origin-centered dimensions. Proof for all propositions below are included in Appendix A.

## 2.1 Dimension Regularization for Spectral Embeddings

In the case of Adjacency Spectral Embeddings (ASE) [6], which are equivalent to taking the leading eigenvectors of the adjacency matrix, the matrix  $S$  is the adjacency matrix  $A \in \{0, 1\}^{n \times n}$ . For convenience, we define  $P$  and  $N$  for an individual node  $i$ , where the full function simply sums over all nodes (e.g.,  $L(X, S) = \sum_{i \in V} L(X, S, i)$ ):

$$L_{ASE}(X, S, i) = \|S_i - XX_i\|_2^2, \quad (2)$$

$$P_{ASE}(X, S, i) = -2 \sum_{j \in \{j' | S_{ij'}=1\}} X_i^T X_j + \|S\|_F^2, \quad (3)$$

$$N_{ASE}(X, S, i) = \|XX_i\|_2^2. \quad (4)$$

On one hand, performing gradient descent on  $N_{ASE}$  can be interpreted as repelling all pairs of embeddings where the repulsion magnitude is the dot product between embeddings. If  $\eta$  is a learning rate and  $t$  is the step count, the embedding for node  $i$  is updated as:

$$X_i^{t+1} = X_i^t - 2\eta \sum_{i' \in V} (X_i^T X_{i'}) X_{i'} \quad \forall i \in V. \quad (5)$$

The same negative function can also be written as a dimension covariance regularization:

**PROPOSITION 2.1.**  $N_{ASE}$  is equivalent to the regularization function  $\|X^T X\|_F^2$  which penalizes covariance among dimensions.

With Proposition 2.1, we can re-interpret the gradient descent updates in Equation (5) as collectively repelling dimensions. The

gradient update can now be written in terms of dimensions:

$$X_{.j}^{t+1} = X_{.j}^t - 2\eta \sum_{j' \in [d]} (X_{.j}^T X_{.j'}) X_{.j'} \quad \forall j \in [d]. \quad (6)$$

## 2.2 Dimension Regularization for Skip-Gram Embeddings

We now introduce a dimension-based approach for skip-gram embeddings. For skip-gram embeddings, the similarity matrix is defined such that  $S_{ij} = 1$  if node  $j$  is in the *neighborhood* of  $i$ . For first-order LINE, the neighborhood for node  $i$  is simply all nodes connected to  $i$  whereas for node2vec, the neighborhood is defined as all nodes reached via a biased random walk initiated at  $i$ . The skip-gram (SG) loss functions can be decomposed as:

$$L_{SG}(X, S, i) = - \sum_{j \in V} S_{ij} \log \sigma(X_i^T X_j) + (1 - S_{ij}) \log \sigma(-X_i^T X_j), \quad (7)$$

$$P_{SG}(X, S, i) = - \sum_{j \in \{j' | S_{ij'}=1\}} \log \sigma(X_i^T X_j), \quad (8)$$

$$N_{SG}(X, S, i) = - \sum_{j \in \{j' | S_{ij'}=0\}} \log \sigma(-X_i^T X_j). \quad (9)$$

Our goal is to map  $N_{SG}$  to a dimension regularization. Recall that this work is motivated by the fact that the purpose of  $N_{SG}$  is to prevent the similarity  $\sigma(X_i^T X_j)$  from increasing for all  $i, j$ ; without  $N_{SG}$ , trivial embedding solutions can emerge that maximize similarity for all pairs of nodes, not just similar pairs.

To measure the onset of the degenerate condition in which all pairs of nodes are similar, we define the *constriction*  $C$  of a set of embeddings to be the minimum dot product between any pair of nodes:

**Definition 2.1 (Constriction).** The constriction  $C \in (0, 1)$  of an embedding matrix  $X$  is defined as:  $C = \min_{i, j \in n \times n} \sigma(X_i^T X_j)$ .

Geometrically, the embedding constriction is maximized when embeddings are radially squeezed and growing in magnitude, that is, collapsed. Proposition 2.2 states that if we remove  $N_{SG}$  altogether and the embeddings are initialized with sufficiently small norm and learning rate, the degenerate collapse will inevitably arise during the course of gradient descent. In the context of graph neural networks, Proposition 2.2 provides conditions under which embedding oversmoothing is guaranteed. In Appendix A.2.1, we validate Proposition 2.2 by showing that embeddings inevitably collapse if only attraction updates are applied for various synthetic sparse graphs.

**PROPOSITION 2.2.** *As the Euclidean norm of the initial embeddings and the learning rate approach zero, then for any constriction threshold  $c \in (0, \infty)$  there exists a time step  $t$  such that after applying gradient descent on  $P_{SG}$  for  $t$  epochs the constriction  $C \geq c$ .*

The proof sketch for Proposition 2.2 is as follows: as the embeddings are initialized closer to the origin, gradient descent on  $P_{SG}$  approaches gradient descent on the matrix completion loss function:  $\|\mathbf{1}_{S>0} \odot (\bar{\mathbf{1}}\bar{\mathbf{1}}^T - \frac{1}{2}XX^T)\|_F^2$ , where  $\mathbf{1}$  is the indicator matrix for whether entries of  $S$  are positive. From Gunasekar et al. [10],

gradient descent implicitly regularizes matrix completion to converge to the minimum nuclear solution; this implicit regularization drives all dot products to be positive, not just pairs of embeddings corresponding to connected nodes.

Now, we show that as constriction increases, performing gradient descent on  $N_{SG}$  approaches optimizing a dimension regularizer. That is, when repulsion is most needed and the embeddings approach collapse due to similarity preservation, repulsion can be achieved via regularization.

First, we map  $N_{SG}$  to an “all-to-all” node repulsion. While  $N_{SG}$  only sums over negative node pairs ( $i, j$  where  $j$  is *not* in the neighborhood of  $i$ ), for large, sparse graphs we can approximate  $N_{SG}$  with the objective  $N'_{SG}$  which sums over all pairs of nodes:

$$N'_{SG} = - \sum_{i,j} \log \sigma \left( -X_i^T X_j \right). \quad (10)$$

Proposition 2.3 states that if the embedding norms are bounded and the constriction  $C > 0$ , then, in the limit of  $n$ , the difference between the gradient of  $N'_{SG}$  and  $N_{SG}$  approaches zero.

**PROPOSITION 2.3.** *If all embeddings have norms that are neither infinitely large or vanishingly small and the embedding constriction  $C > 1/2$ , then, as the number of nodes in a sparse graph grows to infinity, the gradients of  $\nabla N_{SG}$  and  $\nabla N'_{SG}$  converge:*

$$\lim_{n \rightarrow \infty} \frac{\|\nabla N'_{SG} - \nabla N_{SG}\|_F^2}{\|\nabla N_{SG}\|_F^2} = 0, \quad (11)$$

where a graph is sparse if  $|E|$  is  $\mathcal{O}(n^2)$ .

For a single node  $i$ , performing gradient descent on Equation (10) results in the following update:

$$X_i^{t+1} = \left( 1 - \eta \sigma \left( \|X_i^t\|^2 \right) \right) X_i^t - \eta \sum_{i' \in V} \sigma \left( (X_i^t)^T X_{i'}^t \right) X_{i'}^t. \quad (12)$$

The right-hand term in the gradient update repels node  $i$  from all other nodes where the repulsion is proportional to the similarity between the node embeddings. In Proposition 2.4, we show a connection between minimizing  $N'_{SG}$  and centering the dimensions at the origin. For intuition, observe that if all pairs of nodes are highly similar, the gradient update in Equation (12) is approximately equal to subtracting the column means scaled by a constant ( $2\eta \left( X^T \bar{\mathbf{1}} \right)$ ). This is equivalent to performing gradient descent on a dimension regularizer that penalizes non-zero dimension means,

$$R(X) = \|X^T \bar{\mathbf{1}}\|_2^2. \quad (13)$$

We formalize the connection between the negative function  $N'_{SG}$  and origin-centering in the following proposition:

**PROPOSITION 2.4.** *Let  $R$  be the dimension regularizer defined in Equation (13) that penalizes embeddings centered away from the origin and  $n \gg d$ . Then, as the constriction increases beyond zero, the difference between performing gradient descent on  $R$  versus  $N'_{SG}$  vanishes.*

We note that our result establishing a connection between the skip-gram loss and origin-centered dimensions is analogous to the finding in Wang and Isola [25] connecting the InfNCE loss with embeddings uniformly distributed on unit hyperspheres.

**2.2.1 Comparison with Skip-Gram Negative Sampling.** Skip-gram negative sampling (SGNS) offers an efficient stochastic approximation to the gradient update in Equation (12). Furthermore, the SGNS procedure provides a tunable way to bias the gradients—via non-uniform sampling—in a manner that has been seen to empirically improve the utility of the resulting embedding in downstream tasks [16]. Instead of repelling node  $i$  from all other  $n - 1$  nodes, SGNS repels  $i$  from a sample of  $k$  nodes where the nodes are sampled according to a distribution  $P_\alpha$  over all nodes, optimizing the following objective:

$$N_{SGNS}(X, S, i) = -k \mathbb{E}_{j' \sim P_\alpha} \left[ \log \sigma \left( -X_i^T X_{j'} \right) \right], \quad (14)$$

where the expectation is estimated based on  $k$  samples.

In aggregate, SGNS reduces the gradient time complexity from  $\mathcal{O}(n^2)$  to  $\mathcal{O}(kn)$ . With Proposition 2.4, instead of reducing the number of pairwise embedding repulsions, we focus on dimension operations, reducing the time complexity from linear in  $n$  to linear in  $d$ .

As mentioned, SGNS embeddings can be tuned by the choice of the non-uniform sampling distribution, where in graph embedding contexts the distribution  $P_\alpha$  is typically sampling nodes proportional to their degree $^\alpha$ , with  $\alpha = 3/4$ . An optimization-based intuition for this choice is that a degree-based non-uniform distribution prioritizes learning the embeddings of high-degree nodes, but we emphasize that the specific choice of  $\alpha = 3/4$  is typically motivated directly based on improved empirical performance in downstream tasks.

While beyond the scope of the present work, we briefly note that our dimension regularization framework is immediately amenable to introducing an exactly analogous tuning opportunity. We can simply replace the regularization in Equation (15) with

$$R(X; \vec{p}) = \|X^T \vec{p}\|_2^2, \quad (15)$$

where  $\vec{p}$  is a normalized weight vector that biases the negative update in exact correspondence to the probabilities of each node in  $P_\alpha$ . In our later simulations, we focus our efforts on the uniform case of  $\vec{p} = \bar{\mathbf{1}}$ , i.e., the regularizer in Equation (15).

### 3 ALGORITHM AUGMENTATION TO REPLACE SGNS

We now propose a generic algorithm augmentation framework that directly replaces SGNS with dimension regularization. We instantiate this algorithm augmentation for LINE and node2vec but note that the framework is applicable to any graph embedding algorithm using SGNS.

Our augmentation modifies existing algorithms using SGNS in two ways. First, the augmentation prioritizes the positive function  $P$ , that is preserving similarity when possible. Not only does prioritizing observed edges increase efficiency, but it is also desirable given the fact that real-world graph data frequently have missing edges [18, 31]. We prioritize the observed edges by defaulting to performing gradient descent on  $P$ .

The second augmentation replaces SGNS with gradient descent on  $R$ , a dimension regularizer that induces mean-centered dimensions, as introduced in Equation (13). The regularizer directly replaces repelling nodes embedding from negative samples, and, if

the embeddings exhibit high constriction after repeated gradient updates to  $P$ ,  $\nabla R$  approximates  $\nabla N'_{SG}$  as established in Proposition 2.4.

Taken together, the algorithm augmentation framework can be summarized as:

$$X^{t+1} = \begin{cases} X^t - \eta \nabla P_{SG}(X^t) & t \% n_{\text{negative}} \neq 0, \\ X^t - \eta [\nabla P_{SG}(X^t) + \lambda \nabla R(X^t)] & t \% n_{\text{negative}} = 0, \end{cases} \quad (16)$$

where  $\lambda$  is the regularization hyperparameter,  $n_{\text{negative}}$  controls the frequency of performing gradient descent on  $R$ , and  $\eta$  is the learning rate.

Figure 2 visualizes our algorithm augmentation with a toy stochastic block model, where we compare SGNS and dimension regularization side-by-side. In both cases, we begin by randomly initializing embeddings around the origin and then repeatedly (over 100 epochs) attracting the embeddings of similar nodes. The attraction drives the embeddings away from the origin and causes the two blocks to remain entangled. Then, we apply a repulsion force, using SGNS in the top row and dimension mean regularization in the bottom row, which pulls the embeddings back to the origin. In the case of dimension regularization, once we resume applying positive updates, the embeddings for the two blocks begin to separate, indicating effective repulsion.

### 3.1 Instantiation for LINE and node2vec

In Algorithm 1 we include the pseudo-code for the augmented versions of augmented LINE and node2vec, utilizing the framework in Equation (16). The commented out pseudo-code corresponds to the old negative update code that is removed and replaced by the new negative update code. The below implementation also optimizes the embeddings through batches of positive edges.

---

#### Algorithm 1 Augmented LINE and node2vec

---

```

Input:  $G, n, d, p, q, \text{num\_batches}, \text{batch\_size}, \lambda, \eta, n_{\text{negative}}$ 
 $X^0 \leftarrow \text{random\_initialization}(n, d)$ 
 $walks \leftarrow \text{run\_random\_walks}(G, p, q)$ 
for  $t \in \{1, \dots, \text{num\_batches}\}$  do
   $X^{t+1} \leftarrow X^t$ 
  for  $j \in \{1 \dots \text{batch\_size}\}$  do
     $i, j \leftarrow \text{sample\_uniform\_pair}(walks)$ 
     $X_i^{t+1} \leftarrow X_i^t + \eta \sigma(-\langle X_i^t, X_j^t \rangle) X_j^t$   $\triangleright$  positive update
     $X_j^{t+1} \leftarrow X_j^t + \eta \sigma(-\langle X_i^t, X_j^t \rangle) X_i^t$ 
    for  $j' \in \text{sample}(P_\alpha, k)$  do  $\triangleright$  old negative update
       $X_i^{t+1} \leftarrow X_i^t - \eta \sigma(-\langle X_i^t, X_{j'}^t \rangle) X_{j'}^t$ 
       $X_{j'}^{t+1} \leftarrow X_{j'}^t - \eta \sigma(-\langle X_i^t, X_{j'}^t \rangle) X_i^t$ 
    end for
  end for
  if  $\text{idx} \% n_{\text{negative}} == 0$  then  $\triangleright$  new negative update
    for  $j \in \{1, \dots, d\}$  do
       $X_j^{t+1} \leftarrow X_j^t - \frac{\lambda}{n} \sum_i X_{ij} \vec{1}$ 
    end for
  end if
end for

```

---

LINE and node2vec differ in their implementation of the random-walk generation function. For LINE, the function simply returns the edge set  $E$ . For node2vec, the function returns the set of all node and neighbor pairs ( $\{i, j | S_{ij} = 1\}$ ) where neighbors of  $i$  are nodes encountered on a biased random walk starting at  $i$ . The parameters  $p, q$  control the bias of the random walk, as specified in Grover and Leskovec [9].

## 4 EVALUATION

In this section, we present our evaluation of augmented LINE and augmented node2vec where SGNS has been replaced with dimension regularization via Algorithm 1. We show that on a variety of real-world networks, the augmented algorithms achieve better link-prediction results while also reducing training runtime. We also compare the augmented algorithms to baselines in which the negative function has been removed completely; while these baselines perform well on real-world networks, we show that they rely on low levels of connectivity to sustain link-prediction performance.

### 4.1 Evaluation Methodology

We evaluate the augmented algorithms on a downstream link-prediction task where the two evaluation metrics are Precision@ $k$  and Transductive Precision@ $k$ . Both are detailed below:

*Precision. (In-Sample).* After training an algorithm on a graph  $G$  and obtaining an embedding matrix  $X$ , the precision@ $k$  for a single node  $i$  is the fraction of the top- $k$  nodes most similar to node  $i$  that are connected to  $i$ . Similarity between two nodes is defined as the dot product of their embeddings. The precision for the whole graph is the average node precision for all nodes with degree at least  $k$ . We refer to precision as in-sample precision since it is a measure of how much the embeddings overfit to the edges they are trained on. In all of our experiments, we set  $k = 10$ .

*Transductive Precision. (Out-of-Sample).* We also measure the link-prediction precision for the embeddings on a hold-out set of edges. Before training the embeddings, we uniformly randomly split  $E$  into  $E_{\text{train}}$  and  $E_{\text{test}}$  following a 70 – 30 split. Let  $V'$  be the set of nodes in the largest connected component induced by  $E_{\text{train}}$ . Then, we define train and test graphs  $G_{\text{train}} = (V', E_{\text{train}})$  and  $G_{\text{test}} = (V', E_{\text{test}})$ . The transductive precision is the precision of embeddings trained on  $G_{\text{train}}$  but evaluated on  $G_{\text{test}}$ . Similar to the previous section, we refer to transductive precision as out-of-sample precision because it is a measure of how much embeddings generalize to unseen edges.

*Baselines.* We compare the augmented algorithms against (unaltered) LINE and node2vec, which we call the *Vanilla* baselines. In addition, we compare against baselines in which the negative function has been removed entirely, which we call the *Positive Only* baselines. We use these baselines to demonstrate that the embeddings collapse in practice without a negative function. We also analyze both the unnormalized and normalized *Positive Only* embeddings as past work has shown that normalizing embeddings improves downstream performance [26].

*Hyperparameters.* The two hyperparameters for the augmented algorithms are  $\lambda$ , the multiplicative regularization factor, and  $n_{\text{negative}}$ ,

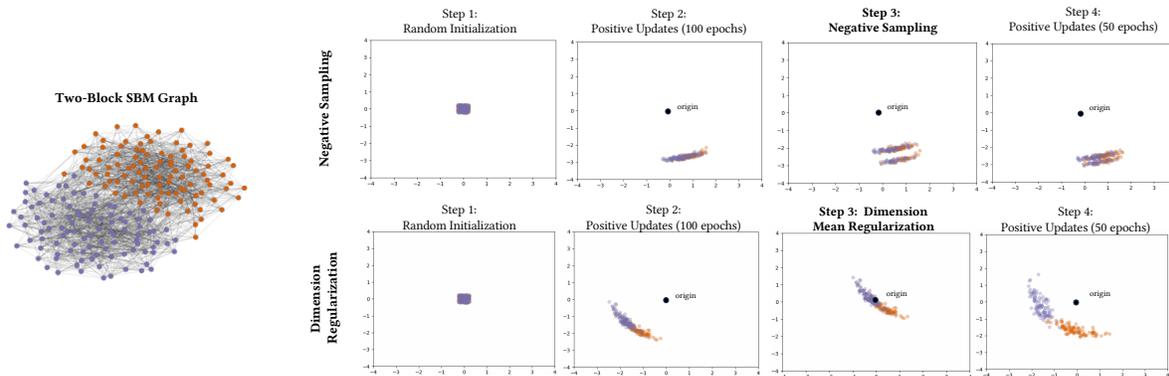


Figure 2: We use a toy two-block SBM example to summarize how we bypass skip-gram negative sampling with dimension regularization. We introduce an algorithm augmentation that prioritizes attracting embeddings of similar nodes together (Step 2), where similarity is the sigmoid of embedding dot products. Eventually, to prevent all pairs of embeddings from becoming similar, the dimension regularization re-centers the embeddings around the origin, increasing node contrast (Step 3). Attraction updates are then again repeatedly applied, but the blocks are more distinguishable post regularization (Step 4).

Table 2: Graph datasets used in our study.

Graph	Type	$n$	$m$
Wikipedia	Language	4.8K	93K
Facebook	Social	4.0K	88K
Protein-Protein	Biological	3.9K	39K
ca-HepTh	Citation	8.6K	25K
LastFM	Social	7.6K	28K

where a regularization update is performed once ever  $n_{\text{negative}}$  epochs. We perform a hyperparameter grid search over  $\lambda \in \{0.5, 1, 5, 25, 50, 75, 100\}$  and  $n_{\text{negative}} \in \{1, 10, 100, 1000\}$ . For each graph and vanilla algorithm, we select the hyperparameter configuration that maximizes the transductive precision. Unless otherwise noted, the augmented algorithm evaluation results all correspond to the optimal hyperparameter configuration, which we provide in Appendix B.4 and obtain from a grid search.

*Datasets.* The datasets used for evaluation are listed in Table 2. We chose five real-world datasets spanning a variety of domains [14]. For all of the graphs, if there are multiple connected components we keep only the largest. The statistics in Table 2 characterize the largest connected component.

## 4.2 Results

Our results answer three fundamental questions:

*Q1: How well do the augmented algorithms perform on link prediction tasks relative to the original algorithms?*

Figure 3 shows that the augmented embeddings, on average, achieve higher out-of-sample transductive precision values than their vanilla counterparts. In contrast, Figure 4 shows that augmented embeddings achieve lower in-sample precision values. The figures suggest that the augmented embeddings better generalize to unseen edges, whereas the vanilla embeddings overfit to the training dataset and struggle to generalize well. A notable case is

the Wikipedia graph, where replacing SGNS with dimension regularization yields a 12-fold increase in transductive precision. The improved performance of dimension regularization can be attributed to reduced overfitting on the training dataset, as witnessed by the fact that on average, in-sample Precision@ $k$  decreased by 36.7% for LINE and 15.4% for node2vec across the five graphs.

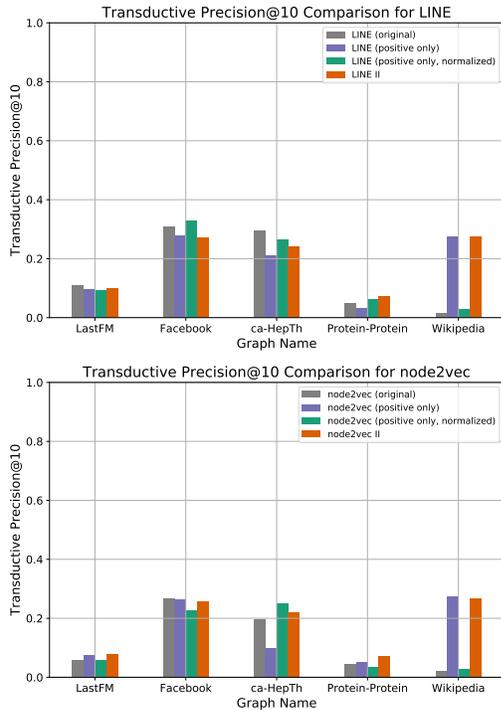
The *Positive Only* baselines perform well both in-sample and out-of-sample at times, but do not do so consistently, as seen in the cases of LastFM and Wikipedia. As we show later in the results section, this inconsistency is due to the strong performance dependence of the *Positive Only* baselines on the underlying graph structure.

*Q2: How much of a runtime speedup does dimension regularization provide over SGNS?*

Figure 5 shows that for both LINE and node2vec, replacing SGNS with dimension regularization significantly reduces runtime while preserving transductive precision. On average, replacing SGNS with dimension regularization decreased the runtime by 64% for LINE and 83% for node2vec. The light orange points in Figure 5 show the runtime and downstream performance for the augmented algorithms across all hyperparameter configurations while the dark orange points denote the best-performing configurations. As expected, completely removing the negative function results in the lowest runtime; thus the *Positive Only* baselines serve as lower bounds on the augmented algorithms’ runtimes. For the node2vec runtime comparison, we use code provided in Grover and Leskovec [9] to generate the random walks and implement batched gradient descent in Python, as detailed in Algorithm 1 to create a level playing field with node2vec II. In practice, an optimized C implementation of our method would be even faster, much like node2vec’s speedup of the gradient descent step achieved via the gensim package. We include additional implementation details in Appendix B.2.

*Q3: For which types of graphs does dimension regularization outperform all baselines?*

We further investigate the limits of completely removing the negative function altogether and show that the inconsistency of

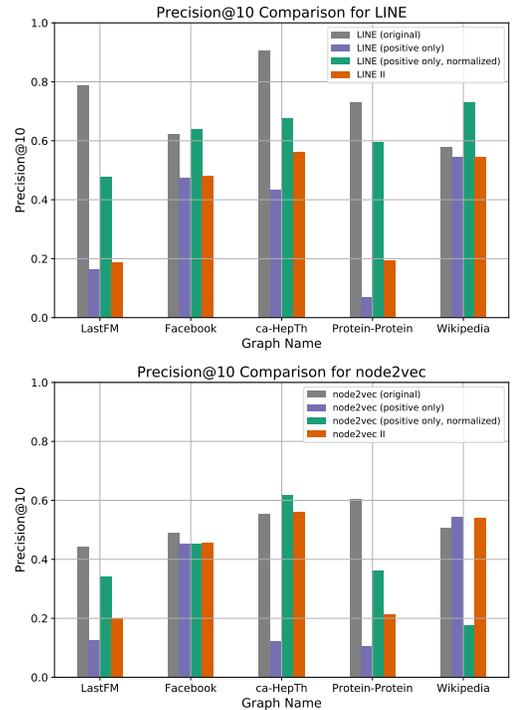


**Figure 3: On unseen edges, the augmented algorithms (LINE II and node2vec II) yield transductive precisions comparable to that of the original algorithms, exceeding in certain cases. This improvement is seen most clearly in the cases of the Protein-Protein, Wikipedia, and LastFM graphs. The Positive Only Normalized baseline performs quite well but the performance relative to the baseline is not consistent, such as in the case of Protein-Protein and Wikipedia.**

the *Positive Only* baselines stems from its performance dependence on graph connectivity. We hypothesize that when the graph has low connectivity the initialization randomness allows nodes to be embedded near neighbors without collectively collapsing. At high connectivity however, collapse occurs.

We construct an experiment in which we measure transductive precision as the between-block edge probability in a Stochastic Block Model (SBM) is gradually increased. We define a two-block SBM where  $n = 200$  and the within-block edge probability  $p$  is 0.2. Then, we gradually increase the between-group edge probability  $q$  until  $p = q$ .

Figure 6 shows that for low values of  $q$  relative to  $p$ , all of the algorithms are able to outperform a random baseline where top- $k$  predictions are uniform samples over  $V$ . When  $q/p$  and graph connectivity is low, the blocks are more distinguishable and link-prediction is an easier task. However, as  $q/p$  increases, the *Positive Only* performance quickly drops off whereas the augmented algorithm performance is more robust. We conclude that as graph connectivity increases, the *Positive Only* baseline is more prone to collapse. We do see that normalization does mitigate the collapse.



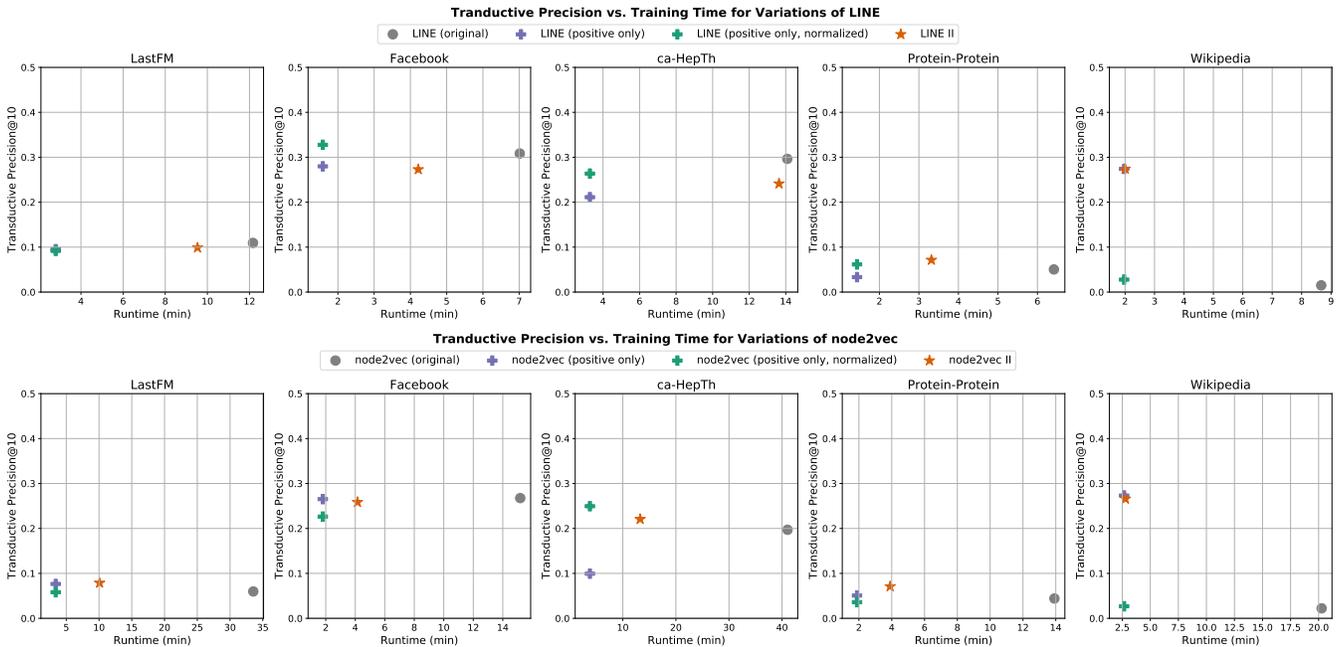
**Figure 4: The above figure shows that the augmented algorithms, on average, perform less well on in-sample precision than the original LINE and node2vec algorithms. One explanation is that the dimension regularization prevents overfitting on the known edges. The Positive Only Normalized baseline performs well on in-sample precision because the algorithm exclusively trains on edge data and the normalization prevents the embeddings from collapsing.**

Finally, as expected, when  $p = q$ , none of the embeddings outperform the random baseline as the SBM becomes an Erdős-Rényi graph.

In Appendix C, we provide more fine-grained empirical evaluation which supplements the results shown here. Namely, analyze whether replacing SGNS with dimensionality reduction benefits or harms specific types of nodes and show that there are not discernable correlations between node feature and link-prediction performance changes. We also demonstrate that our regularization is capable of re-weighting nodes, just as in SGNS, as discussed in (15).

## 5 RELATED WORKS

In this section, we review the popular use of SGNS within graph embeddings and distinguish SGNS from the recent body of literature on negative sampling for self-supervised learning. Our algorithm augmentation is similar in spirit to the growing body of literature on non-contrastive learning, which we also review below.



**Figure 5: The above figure shows the training time and link-prediction performance for each algorithm on each graph. Compared to both vanilla LINE and node2vec (grey circle), our augmentation (orange star) maintains transductive precision while greatly reducing runtime. The results for our augmentation correspond with the optimal hyperparameter configuration; In Appendix B.2 we include runtime results for all configurations as well as details of our runtime evaluation methodology.**

## 5.1 Skip-Gram Negative Sampling

SGNS was introduced in word2vec by Mikolov et al. [16] as an efficient method for learning word embeddings. While the softmax normalization constant is costly to optimize, Mikolov et al. [16] modeled SGNS after Noise Contrastive Estimation (NCE) which learns to separate positive samples from samples drawn from a noise distribution. SGNS has since been adopted for graph representation learning where it is utilized in both unsupervised [9, 19, 24] and supervised skip-gram models [27].

In practice, the negative samples are drawn proportional to some function of node degrees, where many works raise the degree to the  $3/4$  power [9]. Yang et al. [29] study the proper noise distribution in the graph setting and propose sampling negative samples to balance downstream performance while minimizing the Noise Contrastive Estimation loss. This results in negative sample distributions that are positively but sub-linearly correlated with the positive sample distribution.

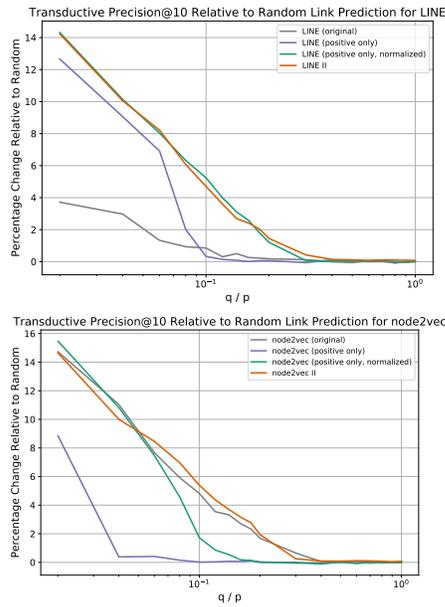
At the same time, there are many known limitations of SGNS. Rudolph et al. [21] place SGNS embeddings within the framework of Exponential Family Embeddings and note that SGNS downweights the magnitude of the negative update and leads to biased embeddings, relative the gradients of the non-sampled objective. Second, Davison and Austern [7] examine the limiting distribution of embeddings learned via SGNS and show that the distribution decouples from the true sampling distribution in the limit. Last, it has also been shown that the optimal noise distribution and the corresponding parameters can vary by dataset [29].

We would also like to note that while the motivations are similar, SGNS differs from the negative sampling that has arisen in the self-supervised learning literature [20]. In self-supervised learning, the negative samples are generally other nodes in the training batch.

## 5.2 Non-contrastive Self-Supervised Learning

Energy Based Models in self-supervised learning are a unified framework for balancing similarity and dissimilarity [13]. As in our decomposition, energy-based models ensure that similar pairs have low energy and dissimilar pairs have high energy. Within energy-based models, there has been more focus across both vision and graph representation learning on contrastive models, which explicitly repel dissimilar pairs [11, 15, 30, 32]. However, given the computational complexities of pairwise contrast, there is a growing body of work on non-contrastive graph representation learning methods [22, 23]. Non-contrastive methods only use positive samples but then regularize to enforce dissimilarity. Shiao et al. [22] show that in certain cases, non-contrastive graph representation learning methods achieve comparable downstream performance while greatly reducing training time.

Recent works have shown that certain non-contrastive methods are simply efficient approaches to performing contrastive learning [8]. That said, these works specifically analyze the squared-loss term and there are no existing works, to our knowledge, establishing a connection between skip-gram loss and non-contrastive methods.



**Figure 6: While removing node repulsion altogether occasionally performed well for the real-world graphs, the *Positive Only* baselines rely on low graph connectivity to perform well. We construct a two-block SBM experiment in which the ratio of the between-block edge probability  $q$  and the within-block probability  $p$  is gradually increased. The figure above shows transductive precision relative to a uniform random baseline. When  $q/p$  is low, the blocks are well-separated and link-prediction performance is high whereas when  $p = q$ , we expect no difference between any algorithm and the uniform-random baseline. In between, the augmented algorithm is most robust to increases in graph connectivity.**

## 6 CONCLUSION

In this work, we provide a new perspective on dissimilarity preservation in graph representation learning and show that dissimilarity preservation can be achieved via dimension regularization. Our main theoretical finding shows that when node repulsion is most needed and embedding dot products are all increasing, the difference between the original skip-gram dissimilarity loss and a regularizer that induces origin-centered dimensions vanishes. Combined with the efficiency of dimension operations over node repulsions, dimension regularization bypasses the need for SGNS. We then introduce a generic algorithm augmentation that prioritizes positive updates, given that real-world graph data often contain missing edges [31], and when node repulsion is needed, utilizes dimension regularization instead of SGNS. Our experimental results show that the augmented versions of LINE and node2vec preserve the link-prediction performance of the original algorithms while reducing runtime by over 60%. In fact, for several real-world graphs, removing dissimilarity preservation altogether performs well; however for synthetic graphs with high connectivity, our augmentation clearly outperforms baselines.

## REFERENCES

- [1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. 2013. Distributed large-scale natural graph factorization. In *WWW'13*. ACM, New York, NY, USA, 37–48.
- [2] Adrien Bardes, Jean Ponce, and Yann LeCun. 2022. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. In *ICLR'22*.
- [3] Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural Graph Learning: Training Neural Networks Using Graphs. In *WSDM'18*. ACM, New York, NY, USA, 64–71.
- [4] Jan Niklas Böhm, Philipp Berens, and Dmitry Kobak. 2022. Attraction-Repulsion Spectrum in Neighbor Embeddings. *JMLR* 23, 95 (2022), 1–32.
- [5] Ines Chami, Sami Abu-El-Hajja, Bryan Perozzi, Christopher Ré, and Kevin Murphy. 2022. Machine Learning on Graphs: A Model and Comprehensive Taxonomy. *JMLR* 23, 89 (2022), 1–64.
- [6] Donniell E. Fishkind Daniel L. Sussman, Minh Tang and Carey E. Priebe. 2012. A Consistent Adjacency Spectral Embedding for Stochastic Blockmodel Graphs. *J. Amer. Statist. Assoc.* 107, 499 (2012), 1119–1128.
- [7] Andrew Davison and Morgane Austern. 2023. Asymptotics of Network Embeddings Learned via Subsampling. *JMLR* 24, 138 (2023), 1–120.
- [8] Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann LeCun. 2022. On the duality between contrastive and non-contrastive self-supervised learning. In *ICLR'23*.
- [9] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *KDD*. Association for Computing Machinery, New York, NY, USA, 855–864.
- [10] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. 2017. Implicit regularization in matrix factorization. In *NIPS'17* (Long Beach, California, USA). Curran Associates Inc., Red Hook, NY, USA, 6152–6160.
- [11] Yeonjun In, Kanghoon Yoon, and Chanyoung Park. 2023. Similarity Preserving Adversarial Graph Contrastive Learning. In *KDD'23*. ACM, New York, NY, USA, 867–878.
- [12] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. 2022. Understanding Dimensional Collapse in Contrastive Self-supervised Learning. In *ICLR'23*.
- [13] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. 2007. Energy-Based Models. In *Predicting Structured Data*. The MIT Press.
- [14] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>.
- [15] Wen-Zhi Li, Chang-Dong Wang, Hui Xiong, and Jian-Huang Lai. 2023. HomoGCL: Rethinking Homophily in Graph Contrastive Learning. In *KDD'23*. ACM, New York, NY, USA, 1341–1352.
- [16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS'13*. Curran Associates, Inc.
- [17] David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *EMNLP'17*. ACL, Copenhagen, Denmark, 2873–2878.
- [18] M. E. J. Newman. 2018. Network structure from rich but noisy data. *Nature Physics* 14, 6 (June 2018), 542–545.
- [19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *KDD'14*. ACM.
- [20] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2021. Contrastive Learning with Hard Negative Samples. In *ICLR'21*.
- [21] Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential Family Embeddings. In *NIPS'16*. Curran Associates, Inc.
- [22] William Shiao, Zhichun Guo, Tong Zhao, Evangelos E. Papalexakis, Yozen Liu, and Neil Shah. 2023. Link Prediction with Non-Contrastive Learning. In *ICLR'23*.
- [23] William Shiao, Uday Singh Saini, Yozen Liu, Tong Zhao, Neil Shah, and Evangelos E. Papalexakis. 2023. CARL-G: Clustering-Accelerated Representation Learning on Graphs. In *KDD'23*. ACM, New York, NY, USA, 2036–2048.
- [24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-Scale Information Network Embedding. In *WWW*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1067–1077.
- [25] Tongzhou Wang and Phillip Isola. 2022. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML'20*.
- [26] Jiacheng Xu and Greg Durrett. 2018. Spherical Latent Spaces for Stable Variational Autoencoders. In *EMNLP'18*. ACL, Brussels, Belgium, 4503–4513.
- [27] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML'16*. JMLR.org, 40–48.
- [28] Zhen Yang, Ming Ding, Tinglin Huang, Yukuo Cen, Junshuai Song, Bin Xu, Yuxiao Dong, and Jie Tang. 2024. Does Negative Sampling Matter? A Review with Insights into its Theory and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), 1–20.
- [29] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding Negative Sampling in Graph Representation Learning. In *KDD'20*. ACM, New York, NY, USA, 1666–1676.
- [30] Zhen Yang, Tinglin Huang, Ming Ding, Yuxiao Dong, Rex Ying, Yukuo Cen, Yangliao Geng, and Jie Tang. 2023. BatchSampler: Sampling Mini-Batches for Contrastive Learning in Vision, Language, and Graphs. In *KDD'23*. Association for Computing Machinery, New York, NY, USA, 3057–3069.
- [31] Jean-Gabriel Young, George T Cantwell, and M E J Newman. 2021. Bayesian inference of network structure from unreliable data. *Journal of Complex Networks* 8, 6 (03 2021).
- [32] Yifei Zhang, Yankai Chen, Zixing Song, and Irwin King. 2023. Contrastive Cross-scale Graph Knowledge Synergy. In *KDD'23*. ACM, New York, NY, USA, 3422–3433.

## A PROOFS

### A.1 Proof for Proposition 2.1

PROOF. Recall that the Frobenius norm of a matrix is equivalent to the trace of the corresponding Gram matrix:

$$N_{ASE}(X, S) = \|XX^T\|_F^2 \quad (17)$$

$$= \text{Tr} \left( XX^T (XX^T)^T \right) \quad (18)$$

$$= \text{Tr} \left( XX^T XX^T \right) \quad (19)$$

$$= \text{Tr} \left( X^T XX^T X \right) \quad (20)$$

$$= \text{Tr} \left( (X^T X) (X^T X)^T \right) \quad (21)$$

$$= \|X^T X\|_F^2, \quad (22)$$

□

where the 4th line follows from the cyclic property of the trace.

### A.2 Proposition 2.2

PROOF. From gradient descent, we know that  $X_i^T X_j$  increases toward infinity for all  $i, j \in E$ ; however, to show that  $X_i^T X_j$  increases for all  $i, j$ , we show that the cosine similarity for all pairs of embeddings approaches 1. We characterize the embedding dynamics in two phases: in the first phase (alignment), the embeddings are initialized near the origin and then converge in direction; then, in the second phase (asymptotic), the embeddings asymptotically move away from the origin while maintaining alignment.

*Phase 1: alignment.* The gradient update rule for  $P_{SG}$  is:

$$X_i^{t+1} = X_i^t + \eta \sum_{j \in N(i)} \sigma \left( -(X_i^t)^T X_j^t \right) X_j^t \quad (23)$$

Because the embeddings are initialized sufficiently small, the sigmoid function can be approximated linearly via a first-order Taylor expansion:

$$\sigma(z) \approx \frac{1}{2} + \frac{1}{4}z \quad (24)$$

In this case, the update rule becomes:

$$X_i^{t+1} \approx X_i^t + \eta \sum_{j \in N(i)} \left( 1 - \frac{1}{2} \left( (X_i^t)^T X_j^t \right) \right) X_j^t \quad (25)$$

The above gradient is equivalent to performing gradient descent on:

$$P'_{SG} = \|\mathbf{1}_{S>0} \odot \left( \overline{\mathbf{1}\mathbf{1}}^T - \frac{1}{2} XX^T \right)\|_F^2 \quad (26)$$

From Gunasekar et al. [10], gradient descent for matrix completion is implicitly regularized to yield the minimum nuclear norm (lowest rank) stationary point. In the case where  $G$  is connected, minimizing the nuclear norm implies that gradient descent on (26) causes  $XX^T$  to approach  $2\overline{\mathbf{1}\mathbf{1}}^T$ , and thus the dot products between all pairs of embeddings increases and  $C \rightarrow 2$ .

*Phase 2: Asymptotic.* For constriction values  $c \leq 2$ , the proof is complete after Phase 1. On the other hand, for constriction values  $c > 2$ , we can complete the proof by showing that following alignment, characterized as positive dot products for all pairs of embeddings, constriction monotonically increases with each gradient

descent epoch. We show the monotonic increase in embeddings via induction. For any pair of embeddings  $X_i^t$  and  $X_j^t$ , the dot product after a single gradient descent epoch is:

$$\begin{aligned} \langle X_i^{t+1}, X_j^{t+1} \rangle &= \left( X_i^t + \eta \sum_{k \in N(i)} \sigma \left( (X_i^t)^T X_k^t \right) X_k^t \right)^T \\ &\quad \left( X_j^t + \eta \sum_{k \in N(j)} \sigma \left( (X_j^t)^T X_k^t \right) X_k^t \right) \end{aligned} \quad (27)$$

At time step  $t'$ , the constriction  $C > 0$ , thus, once all of the dot products are expanded, we have  $\langle X_i^{t'+1}, X_j^{t'+1} \rangle = \langle X_i^{t'}, X_j^{t'} \rangle + C$ , where  $C > 0$ . Thus the dot product between any pair of embeddings strictly increases at time step  $t' + 1$ . By definition, the constriction remains positive, so by induction, we have shown that for all  $t > t'$ , the constriction is monotonically increasing. □

*A.2.1 Supplemental Figures.* Figure 7 provides a high-level summary of the proof for Proposition 2.2. In the beginning, the embeddings are initialized with norm  $\leq b \leq \sqrt{2}$ . Then, the embeddings converge in direction given that near the origin, the gradient of  $P_{SG}$  is the gradient of a matrix completion problem. Further, gradient descent for matrix completion near the origin is implicitly regularized to yield the lowest rank solution, hence a convergence in the embedding direction. Thereafter, once the dot products between all pairs of nodes are positive in the ‘‘Asymptotic’’ phase,  $C$  becomes monotonically increasing.

We also empirically validate Proposition 2.2 in Figure 8. We initialize Erdős-Rényi graphs ( $n = 100$ ) of various densities, randomly initialize the embeddings around the origin, and then apply the positive-only update rule in (23). Figure 8 shows that even when the edge density is 0.05, the constriction eventually becomes monotonically increasing.

### A.3 Proof for Proposition 2.3

PROOF. Let us define the matrix of embedding similarities as  $K = \sigma(XX^T)$ . Then, the gradient of  $N'_{SG}$  is:

$$\nabla N'_{SG} = 2KX \quad (28)$$

If  $\mathbf{1}_{S=0}$  is the indicator matrix where entry  $i, j$  is one if  $S_{ij} = 0$ , then the gradient of  $\nabla N_{SG}$  is:

$$\nabla N_{SG} = (\mathbf{1}_{S=0} \odot K) X \quad (29)$$

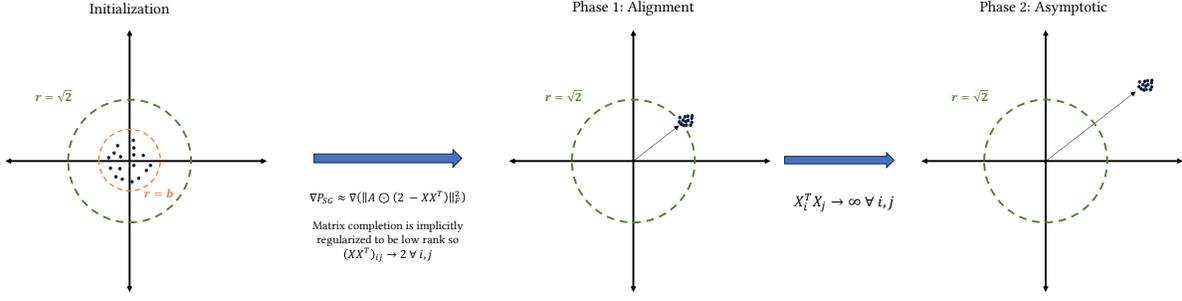
The numerator in the proposition can be upper bounded as:

$$\|\nabla N'_{SG} - N_{SG}\|_F^2 = \sum_{i=1}^n \left\| \sum_{j' \in \{j | S_{ij} > 0\}} \sigma \left( X_i^T X_{j'} \right) X_{j'} \right\|_2^2 \quad (30)$$

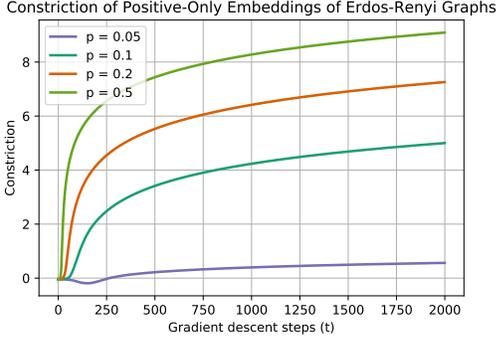
$$\leq \sum_{i=1}^n |\{j | S_{ij} > 0\}| \beta_{\max} \quad (31)$$

$$\leq m \beta_{\max} \quad (32)$$

Where in the above,  $\beta_{\max}$  is a constant and the upper bound on embedding norm squared and  $m$  is the number of non-zero entries in  $S$ .



**Figure 7: High-level overview of proof for Proposition 2.2, which guarantees embedding collapse when only attraction updates are applied. In the beginning the embeddings are initialized near the origin, all with norm at most  $b$ . Then, in Phase 1, attraction update rule is approximately gradient descent for matrix completion; given that the latter is implicitly regularized to yield low-rank solutions the embeddings converge in direction. In the second phase, the embeddings asymptotically distance away from the origin in the same direction.**



**Figure 8: To empirically validate Proposition 2.2, we instantiate Erdős-Rényi networks ( $n = 100$ ) and learn embeddings by only applying the attraction update. The figure shows that for various graph densities, the Constriction eventually becomes monotonically increasing.**

Now we lower bound the denominator. The gradient can be expanded as:

$$\|\nabla N_{SG}\|_F^2 = \sum_{i=1}^n \left\| \sum_{j=1}^n \mathbf{1}_{S_{ij} \neq 0} K_{ij} X_j \right\|_2^2 \quad (33)$$

We can lower bound the norm of the sum by replacing  $X_j$  with the projection of  $X_j$  onto  $X_i$ :

$$\|\nabla N_{SG}\|_F^2 \geq \sum_{i=1}^n \left\| \sum_{j=1}^n \mathbf{1}_{S_{ij} \neq 0} K_{ij} X_j \right\|_2^2 \quad (34)$$

$$\geq \sum_{i=1}^n \left\| \sum_{j=1}^n \mathbf{1}_{S_{ij} \neq 0} K_{ij} \left( \frac{K_{ij}}{\|X_i\|} \right) \left( \frac{X_i}{\|X_i\|} \right) \right\|_2^2 \quad (35)$$

Because the dot product between all pairs of embeddings is assumed to be positive, the norm of the sum is at most the sum of the norms:

$$\|\nabla N_{SG}\|_F^2 \geq \sum_{i=1}^n \sum_{j=1}^n \mathbf{1}_{S_{ij} \neq 0} \left( \frac{K_{ij}}{\|X_i\|} \right)^4 \quad (36)$$

$$\geq \left( \frac{C^2}{\beta_{\max}} \right)^2 (n^2 - m) \quad (37)$$

where above  $m$  is the number of non-zero entries of  $S$ .

Combining the bounds on the numerator and denominator together, we have:

$$\frac{\|\nabla N'_{SG} - \nabla N_{SG}\|_F^2}{\|\nabla N_{SG}\|_F^2} \leq \frac{\beta_{\max}^3 m}{C^4 (n^2 - m)} \quad (38)$$

The left term is a constant given the assumption on constriction and non-vanishing or infinite embedding norms. Further, because the graph is sparse ( $m$  is  $o(n^2)$ ), the second term goes to zero as  $n \rightarrow \infty$ .  $\square$

## A.4 Proposition 2.4

### A.4.1 Lemma for Proof of Proposition 2.4.

LEMMA A.1. Call  $f(x) = \log(1 + \exp(x))$ . We show that both  $f(x) - x \leq \exp(-x)$  and  $|\nabla_x(f(x) - x)| \leq \exp(-x)$ , i.e. have vanishing exponential tails.

PROOF. First, note that  $\log(x) \leq x - 1$ . Since  $e^{-x}(1 + e^x) = 1 + e^{-x}$ , we have,

$$\begin{aligned} \log(e^{-x}(1 + e^x)) &\leq (1 + e^{-x}) - 1 \\ \log(1 + \exp(x)) - x &\leq \exp(-x) \\ f(x) - x &\leq \exp(-x), \end{aligned}$$

where the first line applies the bound on  $\log(x)$  to the equality, the second line organizes terms, and the third applies the definition of  $f(x)$ . Now we have  $\nabla_x(f(x) - x) = \sigma(x) - 1$ . For this,

$$\sigma(x) - 1 = \frac{-1}{1 + \exp(x)} \geq \frac{-1}{\exp(x)} = -\exp(-x).$$

Where the inequality follows reducing the value of the denominator. This concludes the proof.  $\square$

#### A.4.2 Proof for Proposition 2.4.

PROOF. As in the proof for Proposition 2.3, let us define the similarity matrix  $K = \sigma(XX^T)$ .

The gradient for  $N'_{SG}$ , defined in Equation (10), is:

$$\nabla_X N'_{SG} = 2KX \quad (39)$$

The constriction  $C$  is the minimum value of the matrix  $K$ . From lemma A.1, we know that as constriction increases, the difference between 1 and each of the entries of  $K$  vanishes exponentially. Thus, there is a vanishing difference between  $\frac{1}{2}\nabla_X N'_{SG}$  and  $1X$ , where  $1$  is the  $n \times n$  all-ones matrix. Note that  $1X$  is also the gradient of the dimension regularizer  $R$  introduced in (13). Putting these together we have:

$$\left\| \frac{1}{2}\nabla_X N'_{SG} - \nabla R \right\|_2^2 = \left\| (\sigma(XX^T) - 1)X \right\|_2^2 \quad (40)$$

$$\leq \sum_{i=1}^n \left\| \sum_{j=1}^n (\sigma(X_i^T X_j) - 1)X_j \right\|_2^2 \quad (41)$$

$$\leq \sum_{i=1}^n \left( \sum_{j=1}^n (\sigma(X_i^T X_j) - 1)^2 \|X_j\|_2^2 \right) \quad (42)$$

$$\leq \left( \frac{n}{eC} \right)^2 \beta_{\max} \quad (43)$$

In the above,  $\beta_{\max}$  is a constant and corresponds to the maximum embedding norm among all embeddings in  $X$ .

Thus, as constriction  $C$  increases, the difference between the gradient of  $\frac{1}{2}N'_{SG}$  and  $R$  vanishes exponentially. By extension, the difference between gradient descent on  $N'_{SG}$  with a step-size of  $\eta$  and gradient descent on  $R$  with a step-size of  $2\eta$  vanishes with increasing  $C$ .  $\square$

## B EXPERIMENTAL METHODOLOGY

### B.1 Graph Generation and Pre-processing

For all of the graph datasets listed in Table 2, we discarded all components except for the largest connected component. For the SBM graphs used in Figure 6, the number of nodes was fixed at 1000, the number of blocks was fixed at 10, and the in-group edge probability  $p$  was fixed at 0.2.

### B.2 Runtime Evaluation

We perform batched gradient descent for both LINE II and node2vec II. In both cases, we construct batches of positive pairs where each batch consists of 1000 pairs. The number of batches is  $5n$ , where  $n$  is the number of nodes in the graph.

To determine the runtime of vanilla node2vec, we utilized the Grover and Leskovec [9] code to generate the random walks but then performed the positive and negative updates in Python instead of using gensim which is implemented in C. This created a level playing field to compare against our augmentation which is implemented purely in Python. To simulate the random walks, we used the simulate\_walks function in the Grover and Leskovec [9] code,

where we sampled 10 random walks of length 80 for each node in the graph. It is important to note that for all of the precision and transductive precision vanilla node2vec results, we utilized the gensim embeddings generated by the Grover and Leskovec [9] code. We confirmed that the node2vec embeddings we generated for the runtime evaluation yielded similar precision results as the gensim embeddings.

All of our experiments were written in Python and executed on machines with Intel Xeon E5-2690 CPU, 2.60 GHz, 30 MB of cache.

### B.3 Initialization

For almost all algorithms, we uniformly randomly initialize the embeddings  $X^{\text{init}}$  in  $(0, 1/(d))^{n \times d}$ . The only exception is for the *Positive Only* baselines, which are most stable when we initialize the embeddings  $X^{\text{init}}$  uniformly in  $(-1/(2d), 1/(2d))^{n \times d}$ , but ultimately still collapse.

### B.4 Hyperparameters

The two hyperparameters for our algorithm augmentation are the regularization coefficient  $\lambda$  and the epoch parameter  $n_{\text{negative}}$ . We perform a hyperparameter grid search over  $\lambda \in \{0.5, 1, 5, 25, 50, 75, 100\}$  and  $n_{\text{negative}} \in \{1, 10, 100, 1000\}$  and select the values of  $\lambda$  and  $n_{\text{negative}}$  that achieve the highest transductive link-prediction performance, for each graph and original algorithm. Table 3 provides the optimal hyperparameters.

Table 3: Optimal hyperparameters

Graph	Original Algorithm	$\lambda$	$n_{\text{negative}}$
LastFM	node2vec	100	1
LastFM	LINE	100	1
Facebook	node2vec	100	1
Facebook	LINE	100	1
ca-HepTh	node2vec	100	1
ca-HepTh	LINE	100	1
Protein-Protein	node2vec	100	1
Protein-Protein	LINE	100	1
Wikipedia	node2vec	1	100
Wikipedia	LINE	1	100

We note that if the regularization update is performed too often (small  $n_{\text{negative}}$ ) and  $\lambda$  is large, the embeddings can diverge; specifically, instead of re-centering around the origin, the embeddings overshoot the origin and “reflect” about the origin, growing in magnitude each time. In our experiments, the embeddings produced numerical overflows when  $\lambda = 100$  and  $n_{\text{negative}} \leq 10$  for the Facebook and Protein-Protein graphs; the Wikipedia graph embeddings overflowed when  $n_{\text{negative}} = 1$  and  $\lambda = 100$ .

### B.5 Transductive Precision

To calculate the transductive precision metric, we begin by splitting the original graph edge set into training and test edge sets, following a 70/30 split. The training graph is then the largest connected component induced by the training edge set. The embeddings are trained on the training graph and then evaluated on the test graph,

which has the same node set as the training graph. The transductive precision@k value is then the average transductive top-k precision among all nodes with at least one neighbor in the test graph; for nodes with degree less than  $k$  in the test graph, we set  $k = d$ , where  $d$  is the test degree of the node. When calculating the precision for a single node, all edges present in the training set are excluded from the top-k selection process.

## C SUPPLEMENTAL EVALUATION

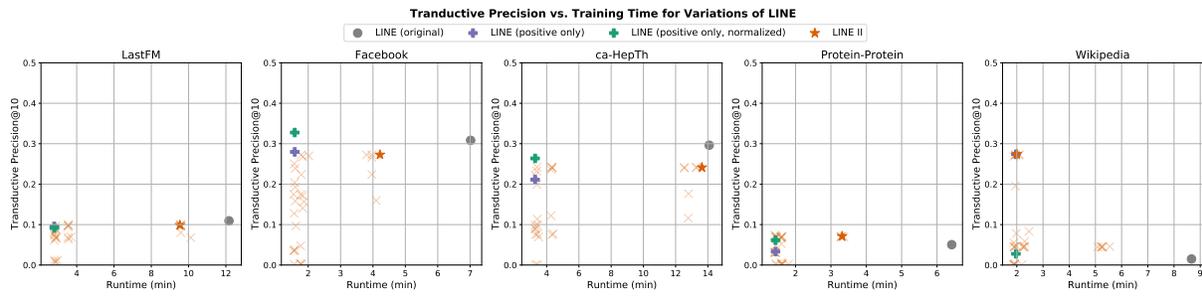
*Runtime vs. Performance.* Figure 5 only shows the augmented algorithm’s results using the optimal hyperparameter configurations, so in Figure 9 we include the runtime results for all hyperparameter configurations, as denoted by the orange Xs. The runtime for our augmentation decreases as  $n_{\text{negative}}$  increases as the regularization update is performed less frequently; when  $n_{\text{negative}} = 1000$ , the runtime is similar to the runtime for the positive-only baselines as nearly only positive updates are provided. For the LastFM, Facebook, ca-HepTh, and Protein-Protein graphs, the optimal hyperparameter configurations set  $n_{\text{negative}} = 1$  so the runtime for the optimal configuration is higher than that for most configurations while remaining less than the vanilla algorithm runtimes. Only for the Wikipedia graph does the optimal configuration yield a runtime comparable to the positive-only baselines.

To supplement the transductive precision results, Figure 10 characterizes all algorithms based on average link-prediction AUC-ROC. The AUC-ROC for a single node is calculated by taking the test-set edges as ground truth labels (1 for neighbors and 0 for non-neighbors) and the dot products between embeddings as prediction scores. The results in Figure 10 confirm the transductive precision results and show that our augmentation reduces runtime while preserving downstream performance.

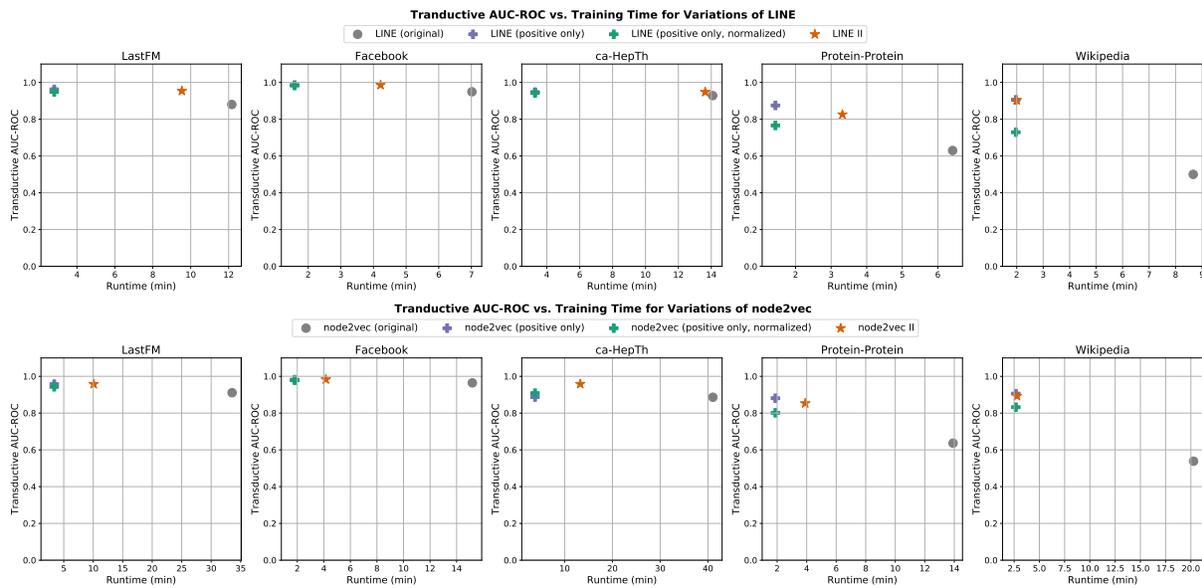
*Performance vs. Node Features.* SGNS traditionally samples nodes with probability proportional to  $d^{3/4}$  where  $d$  is the degree of the node. Because our dimension regularization augmentation defaults to treating nodes uniformly, it is important to assess whether there are correlations between node features and downstream performance. For instance, do low-degree nodes perform better under our augmentation compared to SGNS? Figures 11 and 12 plot downstream performance as a function of node degree, triangle count, and clustering coefficient. Overall, the relationship between node features and downstream performance is quite similar across the algorithms; the main exception, as noted previously, is that for the Wikipedia graph, our augmentation as well as the normalized positive-only baseline perform best. The lack of differences among algorithms is surprising and suggests that for these networks sampling negative samples proportional to  $d^{3/4}$  performs similarly as sampling uniformly.

*Re-weighting Nodes.* As discussed in subsection 2.2.1, our augmentation is also capable of a flexible weighting of nodes when performing repulsion just as SGNS offers this flexibility. While all other empirical evaluations of our augmentation weights nodes uniformly (using (13) over (15)), in Figure 13 we apply a weight of  $d^{3/4}$  to each node as done in traditional SGNS. Figure 13 shows that

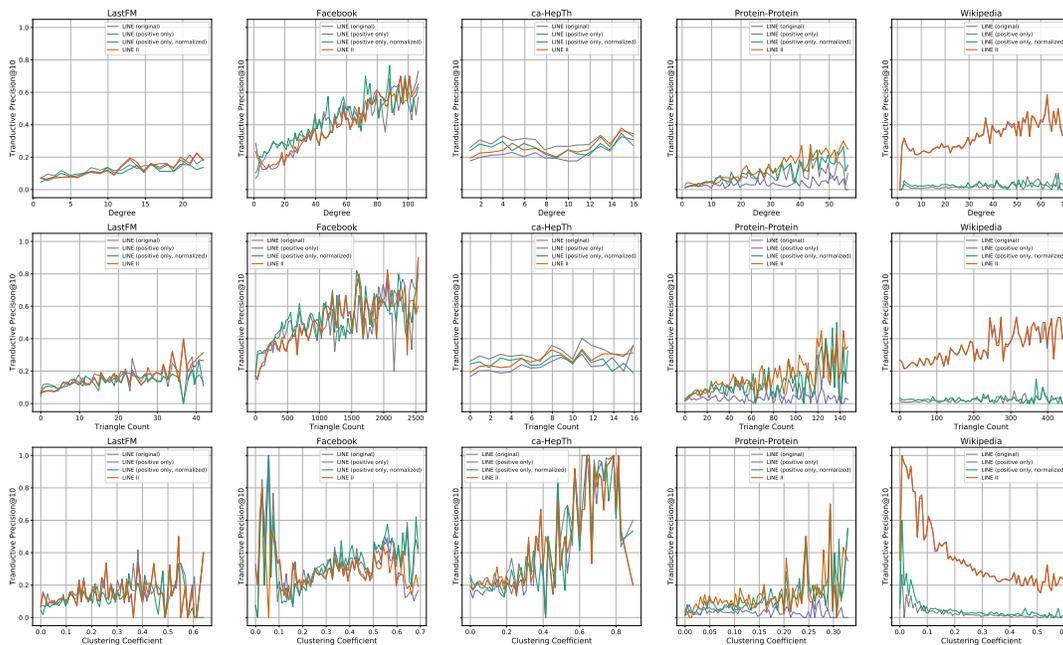
while our augmentation offers flexible re-weighting, the alternative weighting yields results that are very similar to the uniform weighting results shown in the paper.



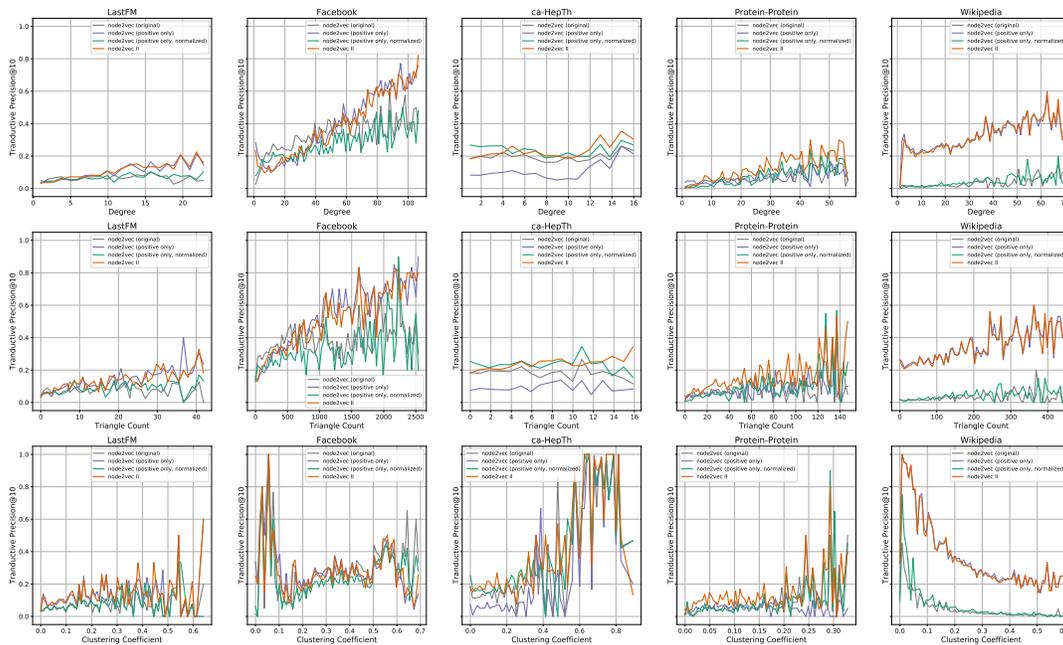
**Figure 9:** To supplement Figure 3, this figure shows the runtime and transductive precision for all hyperparameter configurations of our augmentation algorithm. The optimal configuration is still denoted by the orange star but all other configurations are denoted by orange Xs. The non-optimal hyperparameter configurations yield faster training times but also worse performance.



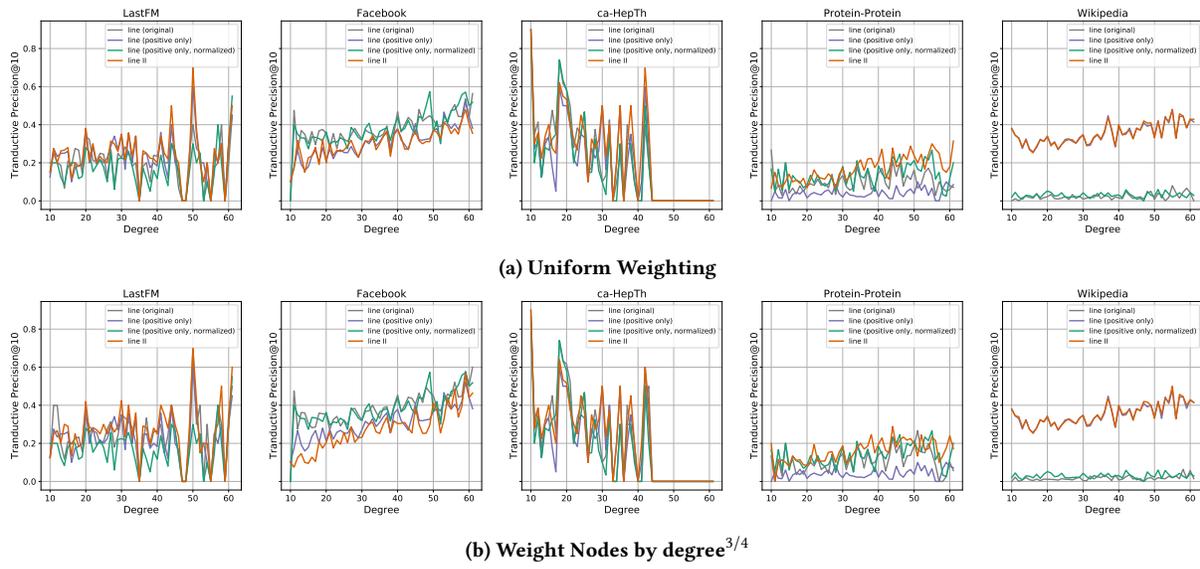
**Figure 10:** The above figure shows the downstream performance of each algorithm as measured by average link-prediction AUC-ROC as opposed to transductive precision. To calculate the AUC-ROC for each node, we take edges in the test set as positive examples and non-edges as negative examples; the prediction scores are the dot products between pairs of embeddings. The AUC-ROC results confirm that our augmentation preserves out-of-sample performance while reducing runtime.



**Figure 11:** Because the SGNS node sampling probability is a function of node degree and our augmentation samples uniformly by default, it is important to assess whether the algorithms differ in their correlations between node features and downstream performance. For each node feature (degree, triangle count, and clustering coefficient), nodes are ranked and divided into 50 bins, and then the average transductive precision is plotted for each bin. Overall, we see that the algorithms share similar correlations between node features and downstream performance despite differences in the node-sampling distributions.



**Figure 12:** The setup for the above figure is analogous with Figure 11 but for node2vec. The takeaways for node2vec are largely the same as those for LINE: the algorithms share similar correlations between node features and downstream performance.



**Figure 13:** While our augmentation is capable of weighting nodes when performing repulsion, just as SGNS weights nodes as a function of degree, applying the traditional SGNS node weights (weight proportional to  $d^{3/4}$ ) via (15) does not significantly impact downstream performance compared to using uniform weights, which we use in all other empirical evaluations. In the figure, the curves for LINE II in the top subfigure correspond to setting  $\vec{p} = \vec{1}$  whereas the LINE II curves in the bottom subfigure correspond to setting the weight of each node to  $d^{3/4}$ , where  $d$  is the node's degree. The curves for all other variations of LINE are identical in the top and bottom subfigures.