

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054

Half-Hop: A graph upsampling approach for slowing down message passing

Anonymous Authors¹

Abstract

Message passing neural networks have shown a lot of success on graph-structured data. However, there are many instances where message passing can lead to over-smoothing or fail when neighboring nodes belong to different classes. In this work, we introduce a simple yet general framework for improving learning in message passing neural networks. Our approach essentially upsamples edges in the original graph by adding “slow nodes” at each edge that can mediate communication between a source and a target node. Our method only modifies the input graph, making it plug-and-play and easy to use with existing models. To understand the benefits of slowing down message passing, we provide theoretical and empirical analyses. We report results on several supervised and self-supervised benchmarks, and show improvements across the board, notably in heterophilic conditions where adjacent nodes may have different labels. We also show how our method can be used to generate multi-scale views for graph self-supervised learning.

1. Introduction

Graph neural networks (GNNs) are now a widely used tool, with applications in recommender systems (Ying et al., 2018), drug discovery (Stokes et al., 2020; Gaudelet et al., 2021), and much more (Monti et al., 2017; Cui et al., 2019; Ktena et al., 2017). However, because graphs are highly variable, building general approaches for learning on graphs that work robustly on many different problems has been a major challenge.

Most GNNs rely on message passing (MP) and aggregation to perform inference (Gilmer et al., 2017). Message passing, while intuitive and simple, can also be limiting in some cases (Alon & Yahav, 2021; Topping et al., 2021; Oono & Suzuki,

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2020). This is especially true when working with complex and heterophilic graphs where nodes from different classes are connected, and in cases where the degree distributions and connectivity is varied throughout the network (Yan et al., 2021). Finding ways to mitigate these problems is of great importance for advancing GNNs, and to do so, we need flexible and generalizable strategies that can easily be applied to both supervised and unsupervised losses.

In this work, we introduce Half-Hop, a simple yet general framework for improving learning in message passing neural networks. The main idea behind our approach is to upsample the input graph: we do this through the introduction of “slow nodes” along each edge that slows-down the messages sent by the source node to the target node. Rather than making explicit modifications of our loss or encoder, we simply modify the input graph, making it plug-and-play and easy to use with existing models and architectures.

We apply our graph upsampling approach to a wide range of benchmark datasets used in supervised and self-supervised node classification tasks. Across the board, we find that our approach provides improvements to the backbones and baseline models that we tested. In self-supervised learning, we demonstrate impressive boosts in performance when applying our approach to state-of-the-art models for self-supervised learning (SSL) (i.e., GRACE (Zhu et al., 2020b), BGRL (Thakoor et al., 2021)). Overall, these results suggest that Half-Hop can significantly improve the performance of GNNs in a wide range of different types of graphs, across models, losses, and tasks.

The contributions of this work include:

- *A graph upsampling approach that improves node classification for a range of GNNs:* In extensive experiments, we show the utility of graph upsampling in both supervised and unsupervised settings.
- *Novel graphs augmentations for SSL:* An important challenge in using SSL on graphs is the design of augmentations to create different views of a graph for contrastive learning (Zhu et al., 2021). We demonstrate that Half-Hop can be used as a standalone augmentation or in conjunction with other augmentations to improve upon state-of-the-art SSL models.
- *Plug-and-play component to improve message passing on heterophilic graphs:* Our results on heterophilic

055
056
057
058
059
060
061
062
063

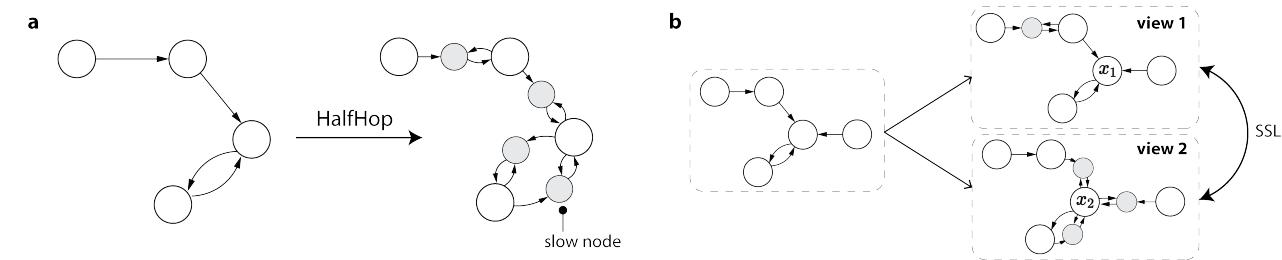


Figure 1. Overview of the Half-Hop augmentation. (a) On the left, we show an original graph and on the right, the graph after applying Half-Hop to all edges. We introduce slow nodes (gray) along each directed edge. (b) Half-Hop is used to generate diverse views for self-supervised learning methods. In the illustrated example, only the incoming edges of randomly selected nodes are half-hopped.

datasets show that by adding Half-Hop to a simple GCN backbone, we can achieve over 10% boost in performance, and when combining Half-hop with GraphSAGE (Hamilton et al., 2017), we achieve results that are comparable with state-of-the-art methods that use more complex architectures.

- *Study of Half-Hop and generalization analysis:* In both a theoretical investigation and empirical studies, we unpack the different ways that Half-Hop impacts learning. **In particular, we analyze the impact of Half-Hop on the dynamics of message passing and show how our approach can slow down the effects of oversmoothing.**

2. Method

In computer vision, increasing the resolution of an image (zooming in) is one of the most widely used augmentations for both supervised and unsupervised learning (Chen et al., 2020; Grill et al., 2020). Our idea is to rescale graphs in a similar fashion. Just as increasing the resolution of an image requires the introduction of new pixels, we upsample graphs by introducing new nodes along edges, and interpolating their node features based on the corresponding source and target nodes (see Figure 1). The modified graph can then be passed into any message passing-based neural network without further modification.

2.1. Background and Notation

Let $G = (V, E)$ denote our graph with nodes V and directed edges E . Each node $v_i \in V$ is associated with a set of d -dimensional features $x_i \in \mathbb{R}^d$. We note directed edge e_{ij} the edge going from node v_i to node v_j . In the message passing scheme, nodes exchange information with their neighboring nodes, through multiple rounds of message passing. Each node v_i updates its embedding by combining their ego-embedding (the node's embedding at a previous step) and the aggregated embeddings received from their neighbors. There are multiple implementations of the message passing layer, we highlight the GCN (Kipf & Welling, 2016) model,

for which the output at layer ℓ is expressed as:

$$h_i^{(\ell)} = \sigma \left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{W}_\ell h_j^{(\ell-1)} \right),$$

where $\mathcal{N}(i)$ denotes the neighbors of v_i , \mathbf{W}_ℓ is a set of learnable shared weights used to compute messages, σ an activation function, and $\hat{d}_i = 1 + |\mathcal{N}(i)|$. Other GNN models include GraphSAGE which uses a different learnable matrix for the ego-embedding, and GAT (Veličković et al., 2017) which leverages attention during the aggregation step.

2.2. Half-Hop: Graph upsampling by inserting slow nodes between adjacent nodes

Half-hopping an edge in the graph. We consider a directed edge e_{ij} that is not a self-loop (i.e. $i \neq j$). To “half-hop” e_{ij} , we introduce a new node v_k that splits the edge into two. The path $v_i \rightarrow v_j$ previously of length 1, is expanded: $v_i \rightarrow v_k \rightleftharpoons v_j$. Due to the added step, we refer to this new node as a “slow node”. This modification to the graph can be expressed as follows:

$$\begin{cases} V' = V \cup \{v_k\} \\ E' = (E \setminus \{e_{ij}\}) \cup \{e_{i \rightarrow k}, e_{j \rightarrow k}, e_{k \rightarrow j}\}, \end{cases} \quad (1)$$

Note that both the source and target nodes communicate their messages to the slow node, but the slow node only passes information in the original direction. We find that this configuration is indeed optimal compared to other connectivity motifs (see ablations in Appendix B).

Interpolating slow node features. When constructing a slow node, we need to decide what features to assign to it, since zero or random initialization is not sufficient (Appendix B). A simple yet effective approach is to use linear interpolation of the source and target features. For edge $e_{i,j}$, we initialize the features of their slow node as follows:

$$\tilde{x}_k = (1 - \alpha)x_j + \alpha x_i,$$

110 where α is some fixed scalar between 0 to 1. This parameter
111 can be fine-tuned using a validation set. By adjusting α
112 between 0 and 1, we can adjust the proximity of the slow
113 node's initial features to the source or target node.
114

115 **Half-hopping the graph.** Now that we have established
116 how an edge is modified, we can apply this operation to
117 multiple edges in the graph. We can do this for all edges
118 or for a subset of them. In our work here, we consider
119 a node-level sampling to apply Half-Hop randomly on a
120 subset of edges. For each node $v_i \in V$, with probability
121 p , we "half-hop" all of v_i 's incoming edges. Let S be the
122 subset of nodes selected for half-hopping, E_S be the set
123 of all directed edges that have target nodes in S , where
124 $N_s = |E_S|$. The set of nodes in the new locally half-hopped
125 graph is: $V' = V \cup \{v_k\}_{k=1}^{N_s}$.

126 To streamline notation, we can write a sample from this
127 graph generative process as $(V', E') \sim hh_\alpha(G; p)$. When
128 *all* edges in the graph are Half-Hopped (i.e. $p=1$), we will
129 use an uppercase $HH_\alpha(G)$. Note that in contrast to the
130 node sampling generator, the fully half-hopped graph is a
131 deterministic transformation.
132

133 2.3. Combining Half-Hop with any MPNN

134 After applying Half-Hop, we obtain a modified graph that
135 can be passed into a message passing neural network. We
136 treat original and slow nodes the same when it comes to
137 MP, both nodes receive and send updates according to the
138 same rules. We treat the Half-Hop operation as internal
139 to the GNN, and any downstream operations such as loss
140 computation or inclusion in further neural network encoders
141 work only over the embeddings of the original nodes.
142

143 After training our model with Half-Hop, at inference time,
144 we have the choice of whether to use the original graph G
145 or the Half-Hopped graph $HH_\alpha(G)$ as input. If we choose
146 to not augment the graph at inference time, we will simply
147 benefit from an improved model generalization, enabled by
148 the use of Half-Hop as an augmentation during training (as
149 we show in Section 4.4). If we do augment the graph, we
150 would also take advantage of the improved MP enabled by
151 Half-Hop (as we demonstrate in Section 3.2).
152

153 2.4. Using Half-Hop to generate views for 154 Self-supervised learning

155 We next consider the use of Half-Hop for self-supervised
156 representation learning. Self-supervised learning methods
157 use augmentations to generate different views of a graph and
158 then learn a representation space in which these different
159 views are close to each other. With Half-Hop, we can create
160 views where node i might be directly connected to its one-
161 hop neighbors in the first, but half-hopped in the second.
162 This generates a heterogeneous zooming and cropping into
163

the neighborhood of different nodes. We expand on this in
164 Section 3.2, where we show how Half-Hop alters the RF of
165 the GNN.
166

167 Thus, we can generate two views $(\tilde{G}_1, \tilde{G}_2)$ of the graph to
168 use as inputs to contrastive learning:
169

$$170 \tilde{G}_1 \sim hh_\alpha(G; p_1), \tilde{G}_2 \sim hh_\alpha(G; p_2)$$

171 where we parameterize the sampling procedure for each
172 view with node-sampling probabilities p_1 and p_2 . As
173 explained in the previous section, when evaluating the
174 contrastive loss, we only use the original nodes as positive/
175 negative examples.
176

3. Understanding Half-Hop

177 In this section, we investigate Half-Hop from two angles.
178 In Section 3.1, we study how Half-Hop alters the receptive
179 field of the GNN model. In Section 3.2.2, we study the effect
180 of Half-Hop on message passing from a spectral perspective.
181 In both theory and experiments, we show that our approach
182 slows down the smoothing process.
183

3.1. Reshaping the node's receptive field with Half-Hop

184 The receptive field (RF) of a model represents the parts of
185 the input graph that have the most significant impact on the
186 final embedding of a particular node. In MPNNs, the repre-
187 sentation of a node is typically influenced by its neighboring
188 nodes. In Half-Hop, the receptive field is reduced because
189 messages are routed through intermediate nodes and thus
190 1-hop neighbors are no longer within 1-hop of one another.
191 Thus, we wanted to understand how the RF is being shaped
192 over rounds of message passing. To do this, we consider a
193 simple 2D planar graph (Figure 2a) over which we apply
194 multiple rounds of mean aggregation.
195

196 In Figure 2b, we visualize, for a given central node in this
197 simple 2D graph, the contribution of a k -hop neighbor to
198 the central node's final embedding. In the original graph
199 (Fig. 2b, left), the weights of messages from nearest neigh-
200 bors are quickly attenuated. Even after one step, the uniform
201 weights between 1-hop neighbors and self-loops flattens the
202 RF immediately from the central node. This smoothing
203 process can often take place extremely quickly, as demon-
204 strated in this example. When we apply Half-Hop (Fig. 2b,
205 right), we show that the receptive field is altered: a more
206 graceful decrease in the weight of neighbors is observed,
207 with a higher amount of weight placed on the node itself.
208 In Figure 2c, we plot the self-weight (y-axis) as a function
209 of the message passing step (x-axis), and show that Half-
210 Hop allows us to preserve the ego-embedding for nodes for
211 longer. Even as we converge to large values of α , Half-Hop
212 preserves a stable value for the self-weight due to the sym-
213 metry in mixing across source and target nodes. At the same
214

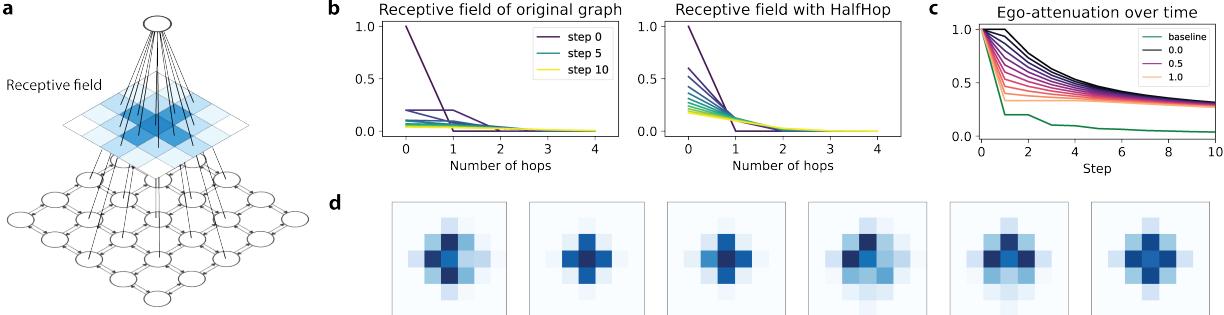


Figure 2. Analyzing how Half-Hop changes the receptive field (RF) of the GNN. Here we consider a 2D planar graph (a) and estimate the contribution of any node to the final embedding of the central node based on distance between the nodes (number of hops), without (b, left) and with Half-Hop (b, right). (c) The dynamics of attenuation of the ego-embedding for different values of α . (d) Example RFs obtained for probabilistic Half-Hop ($\alpha = 0.5, p = 0.75$) that highlight how the RF changes when different subsets of edges are half-hopped.

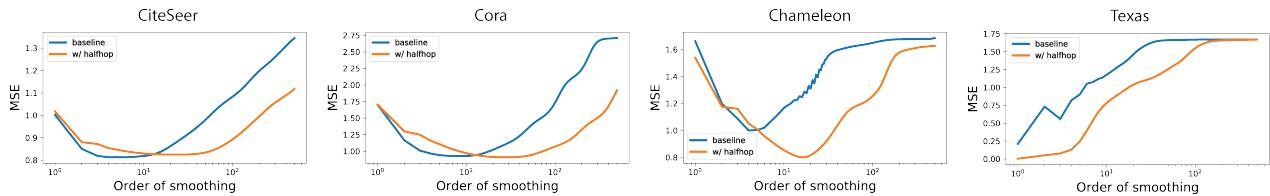


Figure 3. Analysis of isotropic diffusion with and without Half-Hop. Mean Squared Error (MSE) of linear ridge regression on the smoothed features after a number of rounds of message passing. From left to right, we illustrate it for CiteSeer, Cora, Chameleon and Texas.

time, most values of α have similar RFs when considering later stages of MP.

In self-supervised learning, we use Half-Hop to generate multiple views in which the same node can have different receptive fields. In Figure 2c, we highlight a few examples of receptive fields of the same central node when Half-Hop is applied with $\alpha = 0.5$ and $p = 0.75$ on the grid graph. Because we sample a subset of the nodes for half-hopping, we can generate diverse configurations. We find that the receptive field can be narrow, broad or a hybrid of both based on which nodes were half-hopped. This augmentation is reminiscent to zooming and cropping in the vision domain, in that we aggregate information at different scales.

3.2. Half-Hop shapes the dynamics of message passing

The dynamics of message passing depend on the original spectrum of the node features. To provide a rigorous characterization of the dynamics of smoothing, recent work (Keriven, 2022) provides new connections between the performance of GNNs and the covariance of the data. They show that, the covariance play a critical role in determining when over-smoothing starts to occur.

Thus, we hypothesized that by delaying message passing, Half-Hop could both slow down the point where over-smoothing occurs and also alter some properties of the message passing dynamics.

3.2.1. DIFFUSION ON REAL-WORLD DATASETS

To do this, we first use the evaluation framework proposed in (Keriven, 2022) to simulate the effect of diffusion on the Half-Hop graph vs. the original graph for four different real-world datasets (CiteSeer, Cora, Chameleon, Texas). In this case, messages are passed along the graphs using average pooling and no learned weights, and a final output layer is trained from a subset of nodes and tested on those held out unlabeled nodes. On the x-axis, we display the number of rounds of message passing and on the y-axis, the mean squared error (MSE) for the node regression task for all datasets. We note that CiteSeer/Cora are considered homophilic, while Chameleon/Texas are heterophilic.

When we compare the generalization dynamics for Half-Hop vs. the baseline, we find very different behavior for homophilic (left) vs. heterophilic graphs (right). In the homophilic graphs, we achieve overall similar rates of test error with the baseline and Half-Hop but we see the basin of low error solutions is widened (suggesting more stability) and the point where oversmoothing kicks in (MSE starts to go up again) is increased. Naturally, we would expect the factor to be at least twice as large, since Half-Hop divides the receptive field of the graph network by two, but we find that the transition point is even greater than what would be predicted by this factor.

When we examine heterophilic graphs (Chameleon, Texas),

we observe that Half-Hop achieves a significantly lower risk than the baseline. In the case of Chameleon, Half-Hop improves the descent and overall test risk significantly, this can be explained by the fact that heterophilic graphs suffer more from the oversmoothing, since we would be aggregating features of nodes that do not belong to the same class. This demonstrates the useful inductive bias in the Half-Hop model even without learning weights for message passing. In Texas, any rounds of mean aggregation (with learning of weights) hurt the test risk; However, Half-Hop achieves a much lower risk at the first few steps of message passing and also stabilizes the risk far longer. In Appendix F.1, we visualize how the embeddings changes over multiple rounds of MP, and show how Half-Hop also slows down the collapse of the latent space.

3.2.2. ANALYZING THE EFFECT OF HALF-HOP ON GENERALIZATION

Our simulations suggest that Half-Hop changes the dynamics of message passing. Thus, we wanted to dig deeper and develop a result that allows us to compare the dynamics of message passing with and without Half-Hop. Throughout, we follow the assumptions and model described in (Keriven, 2022) (see Appendix A for more details).

A) Preliminaries and assumptions:

Graph Model. We adopt the latent space random graph model, where we assume that the observed node features $x_i = M^\top z_i$ are projections of some underlying latent features z_i , where M denotes an unknown projection matrix; We assume the latent features $z_i \sim \mathcal{N}(0, \Sigma)$, with covariance Σ . The edge weights are determined as a function of the latent variables by $W_{i,j} = \varepsilon + \exp(-\frac{1}{2}\|z_i - z_j\|_2^2)$, where ε is an unknown offset. The node labels, $y_i = z_i^\top \beta^*$, are linear functions of the latent variable z_i with unknown coefficients β^* .

Objective. We consider a semi-supervised linear regression task where the goal is to estimate β^* and use it to predict the labels for nodes in the test set. Because only the projected node features x_i are observed, we can hope to recover the information by leveraging the connectivity of the graph. We use MSE to write the risk as $\mathcal{R}^{(k)} = \|Y_{te} - \hat{Y}_{te}\|^2$, where Y_{te} is stacked labels for n_{te} unlabeled features and \hat{Y}_{te} are the estimated labels after k rounds of message passing.

B) How the spectrum impacts the dynamics of message passing: To present our main result, we first need to define the following function which we will use to approximate our test risk.

Definition 1. (Keriven, 2022) For a symmetric positive semi-definite matrix $S \in R^{d \times d}$, we define the function,

$$R_{\text{reg.}}(S) \stackrel{\text{def.}}{=} \left(\Sigma^{\frac{1}{2}} \beta^*\right)^\top K \left(\Sigma^{\frac{1}{2}} \beta^*\right) \in \mathbb{R}_+$$

where $K = \left(\text{Id} - S^{\frac{1}{2}} M (\gamma \text{Id} + M^\top S M)^{-1} M^\top S^{\frac{1}{2}}\right)^2$, M is the projection matrix, β^* are the true model parameters, and γ is the ridge penalty in our least-squares estimator.

They show that the risk without message passing (MP) can be approximated by $\mathcal{R}^{(0)} \simeq R_{\text{reg.}}(\Sigma)$, and the risk after k rounds of MP can similarly be approximated as $\mathcal{R}^{(k)} \simeq R_{\text{reg.}}(A^{2k} \Sigma)$, where $A = (\text{Id} + \Sigma^{-1})^{-1}$. In this case, we can interpret A as a smoothing operator that is applied to the original spectrum, and interpret $\Sigma^{(k)} = A^{2k} \Sigma$ as the modified covariance after k rounds of MP.

C) Main Result: Our goal is to derive a similar style of result to understand how Half-Hop: (i) impacts the feature covariance of the embeddings, (ii) changes the rate of smoothing as we go deeper and execute more rounds of message passing. To do this, we want to derive an approximation of the risk in the form $R_{\text{reg.}}(\Sigma^{(\text{HH},k)})$ where $\Sigma^{(\text{HH},k)}$ is the approximated covariance of the node features after message passing with Half-Hop. We state our main result below and defer the proof to Appendix A.

Theorem 1. Message Passing Dynamics of Half-Hop. After $k \in \{1, 3, 5, \dots\}$ rounds of message passing, the risk obtained with Half-Hop can be approximated as:

$$\mathcal{R}^{(\text{HH},k)} \simeq R_{\text{reg.}}\left(\frac{1}{2} A^{k-1} \left(\text{Id} + ((1-\alpha)\text{Id} + \alpha A)^2\right) \Sigma\right).$$

The first term $A^{k-1} \Sigma$ smooths the original covariance a rate of $k-1$, which is roughly half the rate of smoothing without Half-Hop ($2k$). This in turn ensures that small eigenvalues decay at a slower rate. The rapid decay of small eigenvalues has been linked to over-smoothing. Half-Hop delays the point at which the smoothing stops being beneficial.

The second term in our augmented covariance, $((1-\alpha)\text{Id} + \alpha A)^2$, reveals the dependence on our mixing parameter α . In particular, we observe a uniform boost of the covariance spectrum coming from the $(1-\alpha)\text{Id}$; this implies that for small values of α we can interpret this as boosting of the self-loops and the minimum eigenvalues in the graph.

4. Experimental Results

In this section, we conduct a comprehensive empirical study of the effectiveness of Half-Hop on a wide range of datasets, models and learning paradigms.

4.1. Experimental Setup

Throughout, we test our approach using three of the most widely used graph models: the Graph Convolutional Network (GCN), GraphSAGE, and Graph Attention Network (GAT). In all of our experiments, we follow the same experimental setup as previous work. In the supervised experiments, we follow (Luo et al., 2022) in splitting the data into a development set and a test set. The hyperparameter

275
276 *Table 1. Increase in supervised performance when using Half-Hop.* The average and standard deviation of accuracy is computed over 20
277 random splits and model initializations. The absolute improvement (Δ) is also reported.

	Am. Comp.	Am. Photos	Co.CS	WikiCS
GCN	90.22 \pm 0.60	93.59 \pm 0.42	94.06 \pm 0.16	81.93 \pm 0.42
HH-GCN	90.92 \pm 0.35	94.52 \pm 0.22	94.71 \pm 0.16	82.57 \pm 0.36
Δ	+0.70 (\uparrow)	+0.93 (\uparrow)	+0.65 (\uparrow)	+0.64 (\uparrow)
GraphSAGE	84.79 \pm 1.08	95.03 \pm 0.33	95.11 \pm 0.10	83.67 \pm 0.45
HH-GraphSAGE	86.60 \pm 0.49	94.55 \pm 0.41	95.13 \pm 0.21	82.81 \pm 0.32
Δ	+1.81 (\uparrow)	-0.48 (\downarrow)	+0.02 (\uparrow)	-0.86 (\downarrow)

286
287 *Table 2. Results on heterophilic graphs.* We report the test accuracy across many datasets, and highlight the absolute improvement (Δ) in
288 classification accuracy when the model is extended with Half-Hop. The “†” results are obtained from (Yan et al., 2021).

	Texas	Wisconsin	Actor	Squirrel	Chameleon	Cornell
Hom level	0.11	0.21	0.22	0.22	0.23	0.30
#Nodes	183	251	7,600	5,201	2,277	183
#Edges	295	466	26,752	198,493	31,421	280
#Classes	5	5	5	5	5	5
GCN†	55.14 \pm 5.16	51.76 \pm 3.06	27.32 \pm 1.10	31.52 \pm 0.71	38.44 \pm 1.92	60.54 \pm 5.30
HH-GCN	71.89 \pm 3.46	79.80 \pm 4.30	35.12 \pm 1.06	47.19 \pm 1.21	60.24 \pm 1.93	63.24 \pm 5.43
Δ	+16.75 (\uparrow)	+19.04 (\uparrow)	+7.80 (\uparrow)	+15.67 (\uparrow)	+21.80 (\uparrow)	+2.70 (\uparrow)
GAT†	52.16 \pm 6.63	49.41 \pm 4.09	27.44 \pm 0.89	36.77 \pm 1.68	48.36 \pm 1.58	61.89 \pm 5.05
HH-GAT	80.54 \pm 4.80	83.53 \pm 3.84	36.70 \pm 0.92	46.35 \pm 1.86	61.12 \pm 1.83	72.70 \pm 4.26
Δ	+28.38 (\uparrow)	+34.12 (\uparrow)	+9.26 (\uparrow)	+9.58 (\uparrow)	+12.75 (\uparrow)	+10.81 (\uparrow)
GraphSAGE†	82.43 \pm 6.14	81.18 \pm 5.56	34.23 \pm 0.99	41.61 \pm 0.74	58.73 \pm 1.68	75.95 \pm 5.01
HH-GraphSAGE	85.95 \pm 6.42	85.88 \pm 3.99	36.82 \pm 0.77	45.25 \pm 1.52	62.98 \pm 3.35	74.60 \pm 6.06
Δ	+3.51 (\uparrow)	+4.70 (\uparrow)	+2.59 (\uparrow)	+3.64 (\uparrow)	+4.25 (\uparrow)	-1.35 (\downarrow)
MixHop†	77.84 \pm 7.73	75.88 \pm 4.90	32.22 \pm 2.34	43.80 \pm 1.48	60.50 \pm 2.53	73.51 \pm 6.34
GGCN†	84.86 \pm 4.55	86.86 \pm 3.29	37.54 \pm 1.56	55.17 \pm 1.58	71.14 \pm 1.84	85.68 \pm 6.63
H ₂ GCN†	84.86 \pm 7.23	87.65 \pm 4.98	35.70 \pm 1.00	36.48 \pm 1.86	60.11 \pm 2.15	82.70 \pm 5.28
MLP†	80.81 \pm 4.75	85.29 \pm 3.31	36.63 \pm 0.70	28.77 \pm 1.56	46.21 \pm 2.99	81.89 \pm 6.40

306
307 tuning is performed using the development set only, and
308 the accuracy of the best model on the development set is
309 reported on the test set. For heterophilic datasets, we use
310 the splits provided by (Pei et al., 2020), and also follow the
311 same hyperparameter search protocol. For self-supervised
312 benchmarks, we use the standard hyperparameters provided
313 for each model and dataset (Zhu et al., 2020b; Thakoor et al.,
314 2021). We provide more details in Appendix C.

316 4.2. Supervised node classification benchmarks

317 In our first set of experiments, we use a set of real-world
318 benchmark datasets – Wiki-CS (Mernyei & Cangea, 2020),
319 Amazon-Computers, Amazon-Photo (McAuley et al., 2015),
320 and Coauthor datasets (Sinha et al., 2015). We train both
321 GCN and GraphSAGE models with and without Half-Hop,
322 and report the results in Table 4. Our results show that
323 Half-Hop provides a good boost in performance across the
324 datasets for the GCN, and GraphSAGE has more variability.
325 HH-GraphSAGE on Amazon Computers is the most
326 significant, where we observe a 1.81% improvement with
327 Half-Hop over the baseline. These results provide evidence
328

329 that Half-Hop can improve learning using only a simple
330 augmentation of the graph.

331 4.3. Heterophilic benchmarks

332 We next applied Half-Hop to common real-world het-
333 erophilic benchmarks, including: Texas, Wisconsin, Actor,
334 Squirrel, Chameleon and Cornell (Luan et al., 2022). On
335 these datasets, we show improvements when we add Half-
336 Hop to GCN, GraphSAGE and GAT. With both HH-GCN
337 and HH-GAT, we see improvement of more than 10-20%
338 on many of the datasets. To place these improvements in
339 the context of more sophisticated methods for heterophilic
340 graphs, we compare with: (i) MixHop (Abu-El-Haija et al.,
341 2019), (ii) GGCN (Yan et al., 2021), and (iii) H₂GCN (Zhu
342 et al., 2020a). See Figure 7 in the Appendix for a compari-
343 son of these different approaches and the techniques used to
344 improve performance on heterophilic graphs.

345 When applied to standard GNNs, we find that Half-Hop
346 boosts performance most significantly for the GCN and
347 GAT, with modest improvements on the GraphSAGE of 2.7% on
348 Cornell and 21.8% improvement on Chameleon. For the

330
331 *Table 3. BGRL with different augmentation.* Performance reported in terms of classification accuracy along with standard deviations. All
332 experiments are performed over 20 random dataset splits and model initializations. At test time, the original graph is used.

	Augmentation	Am. Comp.	Am. Photos	Co.CS	Co.Phy	Wiki-CS
BGRL	None	87.12 ± 0.30	91.18 ± 0.38	91.85 ± 0.25	94.65 ± 0.11	78.69 ± 0.18
	FeatDrop + EdgeDrop	90.34 ± 0.19	93.17 ± 0.30	93.31 ± 0.13	95.73 ± 0.05	79.98 ± 0.10
	GCA	90.39 ± 0.22	93.15 ± 0.37	93.34 ± 0.13	95.62 ± 0.09	–
	Half-Hop	90.47 ± 0.25	93.18 ± 0.26	92.92 ± 0.11	95.69 ± 0.21	79.83 ± 0.53
	FeatDrop + EdgeDrop + Half-Hop	91.02 ± 0.27	93.88 ± 0.19	93.61 ± 0.13	95.75 ± 0.13	80.76 ± 0.71
GRACE	None	77.85 ± 0.96	88.47 ± 0.67	90.04 ± 0.36	OOM	70.61 ± 0.95
	FeatDrop + EdgeDrop	89.53 ± 0.35	92.78 ± 0.45	91.12 ± 0.20	OOM	80.14 ± 0.48
	GCA	87.85 ± 0.31	92.49 ± 0.09	93.10 ± 0.01	OOM	–
	Half-Hop	90.43 ± 0.28	93.58 ± 0.18	92.29 ± 0.12	OOM	79.86 ± 0.41
	FeatDrop + EdgeDrop + Half-Hop	91.11 ± 0.18	94.21 ± 0.26	93.59 ± 0.16	OOM	80.77 ± 0.40

344
345 *Table 4. Increase in performance when using Half-Hop with different SSL frameworks.* Performance reported in terms of classification
346 accuracy along with standard deviations. All experiments are performed over 20 random dataset splits and model initializations.

	input (test)	encoder	Am. Comp.	Am. Photos	Co.CS	Co.Phy	WikiCS
GRACE	G	2-GCN	89.53 ± 0.35	92.78 ± 0.45	91.12 ± 0.20	OOM	80.14 ± 0.48
HH-GRACE	G	2-GCN	91.11 ± 0.18	94.21 ± 0.26	93.59 ± 0.16	OOM	79.77 ± 0.40
HH-GRACE	HH(G)	2-GCN	90.65 ± 0.19	94.89 ± 0.23	94.76 ± 0.14	OOM	80.15 ± 0.16
BGRL	G	2-GCN	90.34 ± 0.19	93.17 ± 0.30	93.31 ± 0.13	95.73 ± 0.05	79.98 ± 0.10
HH-BGRL	G	2-GCN	91.02 ± 0.27	93.88 ± 0.19	93.61 ± 0.13	95.75 ± 0.13	80.76 ± 0.71
HH-BGRL	HH(G)	2-GCN	90.94 ± 0.19	94.50 ± 0.35	94.74 ± 0.15	96.13 ± 0.10	80.37 ± 0.62
BGRL	G	3-GCN	90.04 ± 0.23	92.59 ± 0.34	92.42 ± 0.17	95.32 ± 0.51	78.22 ± 0.77
HH-BGRL	G	3-GCN	90.53 ± 0.27	93.09 ± 0.16	92.58 ± 0.20	95.45 ± 0.09	79.76 ± 0.61
HH-BGRL	HH(G)	3-GCN	91.10 ± 0.21	94.34 ± 0.25	94.76 ± 0.12	96.10 ± 0.09	81.11 ± 0.48

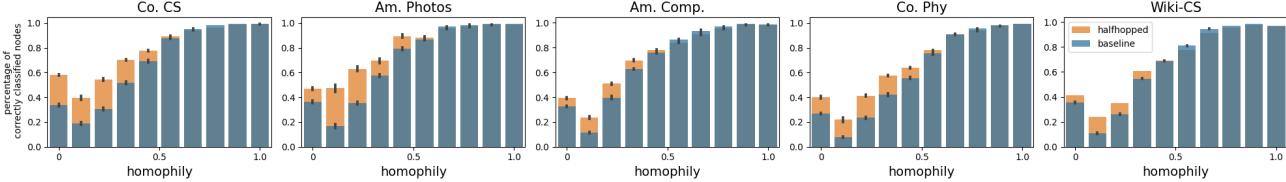
359 GAT, we see even larger improvements on Texas where
360 we get a 28.38% boost with Half-Hop. The GraphSAGE
361 encoder achieves the best performance out of the three
362 models, and HH-GraphSAGE model reaches a performance
363 that is comparable to the other competitor approaches that use
364 more complex model components. We find that our
365 approach provides an impressive gain for simple architectures
366 that rivals with these other models without the need to define
367 complex heuristics or very specialized architectures.
368

4.4. Self-supervised learning benchmarks

370 Graph representation learning methods rely on augmentations
371 that are based on random transformations of the input.
372 Thus, we can test the utility of Half-Hop for creating views
373 for self-supervised learning and also as an add-on with
374 existing augmentations. To do this, we combined Half-Hop
375 with two state-of-the-art methods for self-supervised learning,
376 BGRL (Thakoor et al., 2021) and GRACE (Zhu et al.,
377 2020b). In this case, we report two sets of results: (i) Using
378 Half-Hop as an augmentation during training and then
379 applying the trained model to the original graph G and (ii)
380 Applying Half-Hop to the full graph at inference time.
381

382 **Using Half-Hop to generate views.** First, we examined
383 how Half-Hop could be combined with the existing aug-
384

358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
75100
75101
75102
75103
75104
75105
75106
75107
75108
75109
75110
75111
75112
75113
75114
75115
75116
75117
75118
75119
75120
75121
75122
75123
75124
75125
75126
75127
75128
75129
75130
75131
75132
75133
75134
75135
75136
75137
75138
75139
75140
75141
75142
75143
75144
75145
75146
75147
75148
75149
75150
75151
75152
75153
75154
75155
75156
75157
75158
75159
75160
75161
75162
75163
75164
75165
75166
75167
75168
75169
75170
75171
75172
75173
75174
75175
75176
75177
75178
75179
75180
75181
75182
75183
75184
75185
75186
75187
75188
75189
75190
75191
75192
75193
75194
75195
75196
75197
75198
75199
75200
75201
75202
75203
75204
75205
75206
75207
75208
75209
75210
75211
75212
75213
75214
75215
75216
75217
75218
75219
75220
75221
75222
75223
75224
75225
75226
75227
75228
75229
75230
75231
75232
75233
75234
75235
75236
75237
75238
75239
75240
75241
75242
75243
75244
75245
75246
75247
75248
75249
75250
75251
75252
75253
75254
75255
75256
75257
75258
75259
75260
75261
75262
75263
75264
75265
75266
75267
75268
75269
75270
75271
75272
75273
75274
75275
75276
75277
75278
75279
75280
75281
75282
75283
75284
75285
75286
75287
75288
75289
75290
75291
75292
75293
75294
75295
75296
75297
75298
75299
75300
75301
75302
75303
75304
75305
75306
75307
75308
75309
75310
75311
75312
75313
75314
75315
75316
75317
75318
75319
75320
75321
75322
75323
75324
75325
75326
75327
75328
75329
75330
75331
75332
75333
75334
75335
75336
75337
75338
75339
75340
75341
75342
75343
75344
75345
75346
75347
75348
75349
75350
75351
75352
75353
75354
75355
75356
75357
75358
75359
75360
75361
75362
75363
75364
75365
75366
75367
75368
75369
75370
75371
75372
75373
75374
75375
75376
75377
75378
75379
75380
75381
75382
75383
75384
75385
75386
75387
75388
75389
75390
75391
75392
75393
75394
75395
75396
75397
75398
75399
75400
75401
75402
75403
75404
75405
75406
75407
75408
75409
75410
75411
75412
75413
75414
75415
75416
75417
75418
75419
75420
75421
75422
75423
75424
75425
75426
75427
75428
75429
75430
75431
75432
75433
75434
75435
75436
75437
75438
75439
75440
75441
75442
75443
75444
75445
75446
75447
75448
75449
75450
75451
75452
75453
75454
75455
75456
75457
75458
75459
75460
75461
75462
75463
75464
75465
75466
75467
75468
75469
75470
75471
75472
75473
75474
75475
75476
75477
75478
75479
75480
75481
75482
75483
75484
75485
75486
75487
75488
75489
75490
75491
75492
75493
75494
75495
75496
75497
75498
75499
75500
75501
75502
75503
75504
75505
75506
75507
75508
75509
75510
75511
75512
75513
75514
75515
75516
75517
75518
75519
75520
75521
75522
75523
75524
75525
75526
75527
75528
75529
75530
75531
75532
75533
75534
75535
75536
75537
75538
75539
75540
75541
75542
75543
75544
75545
75546
75547
75548
75549
75550
75551
75552
75553
75554
75555
75556
75557
75558
75559
75560
75561
75562
75563
75564
75565
75566
75567
75568
75569
75570
75571
75572
75573
75574
75575
75576
75577
75578
75579
75580
75581
75582
75583
75584
75585
75586
75587
75588
75589
75590
75591
75592
75593
75594
75595
75596
75597
75598
75599
75600
75601
75602
75603
75604
75605
75606
75607
75608
75609
75610
75611
75612
75613
75614
75615
75616
75617
75618
75619
75620
75621
75622
75623
75624
75625
75626
75627
75628
75629
75630
75631
75632
75633
75634
75635
75636
75637
75638
75639
75640
75641
75642
75643
75644
75645
75646
75647
75648
75649
75650
75651
75652
75653
75654
75655
75656
75657
75658
75659
75660
75661
75662
75663
75664
75665
75666
75667
75668
75669
75670
75671
75672
75673
75674
75675
75676
75677
75678
75679
75680
75681
75682
75683
75684
75685
75686
75687
75688
75689
75690
75691
75692
75693
75694
75695
75696
75697
75698
75699
75700
75701
75702
75703
75704
75705
75706
75707
75708
75709
75710
75711
75712
75713
75714
75715
75716
75717
75718
75719
75720
75721
75722
75723
75724
75725
75726
75727
75728
75729
75730
75731
75732
75733
75734
75735
75736
75737
75738
75739
75740
75741
75742
75743
75744
75745
75746
75747
75748
75749
75750
75751
75752
75753
75754
75755
75756
75757
75758
75759
75760
75761
75762
75763
75764
75765
75766
75767
75768
75769
75770
75771
75772
75773
75774
75775
75776
75777
75778
75779
75780
75781
75782
75783
75784
75785
75786
75787
75788
75789
75790
75791
75792
75793
75794
75795
75796
75797
75798
75799
75800
75801
75802
75803
75804
75805
75806
75807
75808
75809
75810
75811
75812
75813
75814
75815
75816
75817
75818
75819
75820
75821
75822
75823
75824
75825
75826
75827
75828
75829
75830
75831
75832
75833
75834
75835
75836
75837
75838
75839
75840
75841
75842
75843
75844
75845
75846
75847
75848
75849
75850
75851
75852
75853
75854
75855
75856
75857
75858
75859
75860
75861
75862
75863
75864
75865
75866
75867
75868
75869
75870
75871
75872
75873
75874
75875
75876
75877
75878
75879
75880
758



385
386
387
388
389
390
391 *Figure 4. Percentage of correctly classified nodes for a given range of homophily, when using Half-Hop vs. the BGRL baseline.*
392

393 we find a significant enhancement for HH-BGRL on the
394 dataset with the lowest homophily (0.66), WikiCS, with
395 added depth. To better understand the sources of these
396 improvements, we breakdown the node-level accuracies
397 by homophily and show that more heterophilic nodes are
398 classified correctly (Figure 4).

400 5. Related Work

402 5.1. Graph data augmentations for regularization

404 Data augmentation is widely used in graph learning to im-
405 prove the robustness and generalization capabilities of mod-
406 els. Thus, this has spurred a lot of interest in designing
407 augmentations for graph-structured data (Zhu et al., 2021).

408 **Feature perturbations.** The most common feature-based
409 augmentation is feature masking or feature dropout (You
410 et al., 2020; Veličković et al., 2019), which involves ran-
411 domly setting a subset of a node’s features to zero. Other
412 approaches like FLAG (Kong et al., 2022) and LA (Liu et al.,
413 2022) use generative modeling to introduce gradient-based
414 adversarial perturbations to the node’s features. Mixup for
415 graphs has also been proposed but usually requires a par-
416 ticular architecture (Verma et al., 2019), or a sub-graph
417 sampling strategy (Wang et al., 2021).

418 **Edge Perturbation.** Adding and removing edges can also
419 be used to perturb the connectivity of a node. In DropEdge
420 (Rong et al., 2020; Feng et al., 2020; Veličković et al., 2019),
421 each edge has a given probability of being removed. In
422 GCA (Zhu et al., 2021), an edge is more or less likely to
423 be removed based on the connectivity of its target node.
424 GCC (Qiu et al., 2020) uses random walks to sample ego-
425 networks around a central node. Approaches that add edges,
426 on the other hand, typically require more guidance, like
427 GAug (Zhao et al., 2020) which uses neural edge predictors
428 to infer the probability of an edge.

429 **k-Hop augmentations.** Adding edges can also be used
430 to connect more distant neighbors (k -hops away). This is
431 typically achieved using a diffusion process (Li et al., 2019;
432 Hassani & Khasahmadi, 2020). With this type of augmenta-
433 tion, we are effectively expanding the receptive field of the
434 GNN, and are able to replicate the zoom-out operation that
435 we know in images. On the other hand, our proposed aug-
436 mentation, reduces the receptive field and allows a zoom-in
437 operation in that sense.

438 5.2. Graph manipulation at inference time

439 When graphs are sparsely connected, highly heterophilic
440 or have bottlenecks, graph rewiring techniques are used to
441 correct the connectivity of nodes. SDRF (Topping et al.,
442 2021) adds edges based on the Ricci curvature, while DIGL
443 (Gasteiger et al., 2019) uses a diffusion process to add edges.
444 NeuralSparse (Zheng et al., 2020) and GGCN (Yan et al.,
445 2021) are supervised methods that use labels to learn to re-
446 move task-irrelevant edges that typically cause oversmooth-
447 ing in heterophilic neighborhoods. (Ding et al., 2018) uses
448 a generative modeling framework to create a small number
449 of nodes that connect different subgraphs.

450 5.3. Self-supervised and contrastive learning methods

451 Many state-of-the-art approaches for self-supervised learn-
452 ing (SSL) on graphs use augmentations to create different
453 views (i.e., positive examples) and then encourage the repre-
454 sentations of both views to be close in the latent space. For
455 instance, GRACE (Zhu et al., 2020b) uses a contrastive loss
456 to encourage positive examples (a new graph with dropped
457 edges and node features) to become closer to one another,
458 while considering all other nodes to be far away. BGRL
459 instead uses a score-based approach that doesn’t explicitly
460 incorporate negative examples into the loss (Thakoor et al.,
461 2021). Most of these approaches use relatively simple aug-
462 mentations, like node and edge dropout, to create views for
463 learning. However, adaptive graph augmentations like those
464 proposed in GCA (Zhu et al., 2021) that use topology-level
465 and node-attribute-level augmentations like ‘node centrality’
466 measures can also be used to determine edges to mask.

467 6. Discussion

468 In this work, we introduced Half-Hop, a novel network-level
469 augmentation for regularizing learning in graphs. We show
470 that Half-Hop offers significant enhancements in MPNNs,
471 especially in heterophilic conditions and in self-supervised
472 learning. Our results on heterophilic datasets show that
473 by adding Half-Hop to a simple GCN backbone, we can
474 achieve over 10% boost in performance. When we couple
475 Half-Hop with GraphSAGE, we obtain comparable results
476 with state-of-the-art methods that use more complex losses
477 or architectural modifications. In the future, it would be
478 interesting to combine Half-Hop with approaches that use
479 more complex architectures/losses to improve performance
480 further.

References

- References**

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., and Galstyan, A. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pp. 21–29. PMLR, 2019.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=i80OPhOCVH2>.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/chen20j.html>.

Cui, Z., Henrickson, K., Ke, R., and Wang, Y. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4883–4894, 2019.

Ding, M., Tang, J., and Zhang, J. Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 913–922, 2018.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., and Tang, J. Graph random neural networks for semi-supervised learning on graphs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22092–22103. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/fb4c835feb0a65cc39739320d7a51c02-Paper.pdf>.

Gasteiger, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B., Vickers, R., Roberts, C., Tang, J., et al. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6):bbab159, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4116–4126. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/hassani20a.html>.

Keriven, N. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *arXiv preprint arXiv:2205.12156*, 2022.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kong, K., Li, G., Ding, M., Wu, Z., Zhu, C., Ghanem, B., Taylor, G., and Goldstein, T. Robust optimization as data augmentation for large-scale graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 60–69, 2022.

Ktena, S. I., Parisot, S., Ferrante, E., Rajchl, M., Lee, M., Glocker, B., and Rueckert, D. Distance metric learning using graph convolutional networks: Application to functional brain networks. In *Medical Image Computing and Computer Assisted Intervention- MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part I 20*, pp. 469–477. Springer, 2017.

Li, Z., Liu, Z., Huang, J., Tang, G., Duan, Y., Zhang, Z., and Yang, Y. Mv-gcn: Multi-view graph convolutional networks for link prediction. *IEEE Access*, 7:176317–176328, 2019. doi: 10.1109/ACCESS.2019.2957306.

Liu, S., Ying, R., Dong, H., Li, L., Xu, T., Rong, Y., Zhao, P., Huang, J., and Wu, D. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, 2022.

- 495 Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S.,
496 Chang, X.-W., and Precup, D. Revisiting heterophily for
497 graph neural networks, 2022. URL <https://arxiv.org/abs/2210.07606>.
498
- 500 Luo, Y., Luo, G., Yan, K., and Chen, A. Inferring from
501 references with differences for semi-supervised node clas-
502 sification on graphs. *Mathematics*, 10(8), 2022. ISSN
503 2227-7390. doi: 10.3390/math10081262. URL <https://www.mdpi.com/2227-7390/10/8/1262>.
504
- 505 McAuley, J., Targett, C., Shi, Q., and van den Hen-
506 gel, A. Image-based recommendations on styles and
507 substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pp. 43–52,
508 New York, NY, USA, 2015. Association for Computing
509 Machinery. ISBN 9781450336215. doi: 10.1145/2766462.2767755. URL <https://doi.org/10.1145/2766462.2767755>.
510
- 511 Mernyei, P. and Cangea, C. Wiki-cs: A wikipedia-based
512 benchmark for graph neural networks. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.
513
- 514 Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J.,
515 and Bronstein, M. M. Geometric deep learning on graphs
516 and manifolds using mixture model cnns. In *Proceedings
517 of the IEEE conference on computer vision and pattern
518 recognition*, pp. 5115–5124, 2017.
- 519 Oono, K. and Suzuki, T. Graph neural networks exponentially
520 lose expressive power for node classification. In *International Conference on Learning Representations*,
521 2020. URL <https://openreview.net/forum?id=S1ld02EFPr>.
522
- 523 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V.,
524 Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P.,
525 Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E.
526 Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
527
- 528 Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B.
529 Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*,
530 2020. URL <https://openreview.net/forum?id=S1e2agrFvS>.
531
- 532 Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding,
533 M., Wang, K., and Tang, J. Gcc: Graph contrastive
534 coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, pp. 1150–1160, New York, NY, USA, 2020. Association
535 for Computing Machinery. ISBN 9781450379984.
536
- 537 doi: 10.1145/3394486.3403168. URL <https://doi.org/10.1145/3394486.3403168>.
538
- 539 Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge:
540 Towards deep graph convolutional networks on node
541 classification. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=Hkx1qkrKPr>.
542
- 543 Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu,
544 B.-J. P., and Wang, K. An overview of microsoft
545 academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pp. 243–246, New
546 York, NY, USA, 2015. Association for Computing
547 Machinery. ISBN 9781450334730. doi: 10.1145/2740908.2742839. URL <https://doi.org/10.1145/2740908.2742839>.
548
- 549 Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., MacNair, C. R., French, S., Carfrae, L. A., Bloom-Ackermann, Z., et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.
550
- 551 Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer,
552 E. L., Munos, R., Veličković, P., and Valko, M. Large-
553 scale representation learning on graphs via bootstrapping.
554 *arXiv preprint arXiv:2102.06514*, 2021.
555
- 556 Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong,
557 X., and Bronstein, M. M. Understanding over-squashing
558 and bottlenecks on graphs via curvature, 2021. URL
559 <https://arxiv.org/abs/2111.14522>.
560
- 561 Veličković, P., Cucurull, G., Casanova, A., Romero, A.,
562 Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
563
- 564 Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Ben-
565 gio, Y., and Hjelm, R. D. Deep graph infomax. In *International Conference on Learning Representations*,
566 2019. URL <https://openreview.net/forum?id=rklz9iAcKQ>.
567
- 568 Verma, V., Qu, M., Lamb, A., Bengio, Y., Kannala, J.,
569 and Tang, J. Graphmix: Regularized training of graph
570 neural networks for semi-supervised learning. *CoRR*,
571 abs/1909.11715, 2019. URL <http://arxiv.org/abs/1909.11715>.
572
- 573 Wang, Y., Wang, W., Liang, Y., Cai, Y., and Hooi, B. Mixup
574 for node and graph classification. In *Proceedings of the Web Conference 2021*, pp. 3663–3674, 2021.
575
- 576 Yan, Y., Hashemi, M., Swersky, K., Yang, Y., and Koutra,
577 D. Two sides of the same coin: Heterophily and over-
578 smoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*, 2021.
579

550 Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton,
551 W. L., and Leskovec, J. Graph convolutional neural net-
552 works for web-scale recommender systems. In *Pro-
553 ceedings of the 24th ACM SIGKDD international conference
554 on knowledge discovery & data mining*, pp. 974–983,
555 2018.

556 You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen,
557 Y. Graph contrastive learning with augmentations. *Ad-
558 vances in Neural Information Processing Systems*, 33:
559 5812–5823, 2020.

560 Zhao, T., Liu, Y., Neves, L., Woodford, O. J., Jiang, M., and
561 Shah, N. Data augmentation for graph neural networks.
562 *CoRR*, abs/2006.06830, 2020. URL <https://arxiv.org/abs/2006.06830>.

563 Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W.,
564 Chen, H., and Wang, W. Robust graph representation
565 learning via neural sparsification. In III, H. D. and Singh,
566 A. (eds.), *Proceedings of the 37th International Confer-
567 ence on Machine Learning*, volume 119 of *Proceedings
568 of Machine Learning Research*, pp. 11458–11468. PMLR,
569 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/zheng20d.html>.

570 Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and
571 Koutra, D. Beyond homophily in graph neural networks:
572 Current limitations and effective designs. *Advances in
573 Neural Information Processing Systems*, 33:7793–7804,
574 2020a.

575 Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep
576 graph contrastive representation learning, 2020b. URL
577 <https://arxiv.org/abs/2006.04131>.

578 Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L.
579 Graph contrastive learning with adaptive augmentation.
580 In *Proceedings of the Web Conference 2021*, pp. 2069–
581 2080, 2021.

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605 Appendix

606 A. Generalization Analysis

607 A.1. Problem Setup

608 We adopt the model developed in (Keriven, 2022) for our analysis. We invite the reader to refer to their work for more
609 details. In this section, we add the necessary context (assumptions and notations) to set up our own result presented in the
610 section A.2.

611 **Semi-supervised node regression.** We consider semi-supervised learning on an undirected graph of n nodes. We observe
612 the entire graph, encoded by the adjacency matrix W , and the node features x_1, x_2, \dots, x_n . Among the nodes, n_{tr} nodes'
613 labels are observed while $n_{\text{te}} = n - n_{\text{tr}}$ nodes are without labels and used for test. We stack the labels and features as rows
614 of the matrices Y and X , and denote the training labels/features as Y_{tr} and X_{tr} and the testing labels/labels as Y_{te} and X_{te} .

615 Instead of predicting the labels using the node features alone, inference can be improved by leveraging the connectivity
616 of the graph. We consider a linear message passing neural network (MPNN). k rounds of message passing are applied
617 according to the adjacency matrix W , which results in the *smoothing* of the node embeddings. The node labels are then
618 predicted from the updated node embeddings.

619 After multiple rounds of message passing, we then estimate the underlying model parameters from the updated node
620 embeddings as:

$$621 \hat{\beta}^{(k)} = \operatorname{argmin}_{\beta} \frac{1}{2n_{\text{tr}}} \|Y_{\text{tr}} - X_{\text{tr}}^{(k)} \beta\|^2 + \gamma \|\beta\|^2, \quad (2)$$

622 where $X_{\text{tr}}^{(k)}$ are the embeddings of the training nodes obtained after k rounds of message passing, and $\gamma > 0$ is a ridge
623 penalty.

624 The test risk for our semi-supervised setting can be written as

$$625 \mathcal{R}^{(k)} = n_{\text{te}}^{-1} \|Y_{\text{te}} - X_{\text{te}}^{(k)} \hat{\beta}^{(k)}\|^2, \quad (3)$$

626 where $X_{\text{te}}^{(k)}$ are the feature embeddings of the test nodes after k rounds of message passing, and $\hat{\beta}^{(k)}$ is the estimator in (2)
627 learned from the training data.

628 Before we present our analysis, we introduce the following technical function to simplify our later expressions for the risk.

629 **Definition 1. (Keriven, 2022)** For any symmetric positive semi-definite input matrix $S \in R^{d \times d}$, we define the function

$$630 R_{\text{reg.}}(S) \stackrel{\text{def.}}{=} \left(\Sigma^{\frac{1}{2}} \beta^* \right)^{\top} K \left(\Sigma^{\frac{1}{2}} \beta^* \right) \in \mathbb{R}_+$$

631 where $K = (\text{Id} - S^{\frac{1}{2}} M (\gamma \text{Id} + M^{\top} S M)^{-1} M^{\top} S^{\frac{1}{2}})^2$, and M , β^* , Σ are introduced in the model assumptions.

632 **Previous risk results on ordinary MPNN.** In (Keriven, 2022), they show that the risk associated with the ridge regression
633 operator without any message passing of raw features can be approximated by

$$634 \mathcal{R}^{(0)} \simeq R_{\text{reg.}}(\Sigma),$$

635 whereas the risk after k rounds of message passing is approximately,

$$636 \mathcal{R}^{(k)} \simeq R_{\text{reg.}}(A^{2k} \Sigma),$$

637 where $A = (\text{Id} + \Sigma^{-1})^{-1}$.

638 One key implication of this result is that multiple rounds message passing induces a form of spectrum smoothing. This can
639 in some cases, for a small number of steps, induce a helpful form of regularization. As evidenced in above equations, the
640 MPNN, with k rounds of message passing, effectively modifies the key dynamic component of the risk that depends on the
641 covariance of raw data features, from Σ to $A^{2k} \Sigma$.

642 In (Keriven, 2022), they show that analyzing the eigenvalues provides further insight into the smoothing phenomena. We
643 note λ_i an eigenvalue of the covariance matrix Σ , and $\lambda_i^{(k)}$ the eigenvalues after k round of smoothing. (Keriven, 2022)

660 shows that while large eigenvalues mostly maintain their magnitudes $\lambda_i^{(k)} \sim \lambda_i$, small eigenvalues decay exponentially
 661 $\lambda_i^{(k)} \sim \lambda_i^{2k+1}$. In other words, small eigenvalues decay faster than large eigenvalues. In the case where β^* is aligned
 662 with eigenvectors with small eigenvalues, this can introduce harmful bias into the estimator. This framework enables us to
 663 understand how and when smoothing becomes harmful (over-smoothing). The rapid decay of small eigenvalues can be
 664 attributed to this phenomena. In the following, we show that Half-Hop slows down this decay and effectively enables the
 665 more graceful smoothing that we observe empirically in Figure 3.
 666

667 A.2. Main Result

668 To compare the test risk for the original graph vs. when we apply Half-Hop, we use the directed variant of Half-Hop: HH⁽¹⁾
 669 described in Appendix B.1. In this variant, there is no backward edge from the target node to the slow node.

670 **Theorem 1.** The test risk after $k \in \{1, 3, 5, \dots\}$ rounds of message passing with *Half-Hop* are:

$$671 \quad 672 \quad 673 \quad 674 \quad 675 \quad 676 \quad \mathcal{R}_{\text{HH}_\alpha}^{(k)} \simeq R_{\text{reg.}} \left(\frac{1}{2} A^{k-1} \left(\text{Id} + ((1-\alpha)\text{Id} + \alpha A)^2 \right) \Sigma \right),$$

where $A = (\text{Id} + \Sigma^{-1})^{-1}$.

677 Through Theorem 1, we can inspect how Half-Hop changes the rate of smoothing in the graph and can change the underlying
 678 spectral properties of our features. Similar to the original graph, large eigenvalues preserve most of their magnitude through
 679 message passing $\lambda^{(k)} \sim (1 + ((1-\alpha) + \alpha\lambda)^2)\lambda$; However, small eigenvalues decay as $\lambda^{(k)} \sim (1 + (1-\alpha)^2)\lambda^{k+1}$.
 680 From these observations, we see that the decay rate of small eigenvalues is halved compared with ordinary MPNN without
 681 Half-Hop.
 682

683 **Proof.** The basic idea behind our proof is similar to (Keriven, 2022), where we use matrix concentrations to approximate the
 684 node features after multiple rounds of message passing. Following the proof of Theorem 4 in (Keriven, 2022), the regression
 685 risk of $\mathcal{R}_{\text{HH}_\alpha}^{(k)}$ is approximated by $R_{\text{reg.}}(\Sigma')$ where Σ' approximates the covariance of the node features after k rounds of
 686 message passing. Hence, the proof boils down to calculating Σ' .

687 With Half-Hop, we introduce new nodes that we call slow nodes. In the augmented graph, there are two types of nodes: 1.
 688 the original nodes whose features we denote as $x_i \in \mathbb{R}^d$ and 2. the slow nodes, whose features we denote as \tilde{x}_k . When
 689 Half-Hop is applied, the path from original node v_j to original node v_i is replaced by a path from original node v_j to slow
 690 node v_k to original node v_i . We note \mathcal{V}_i the set of (j, k) paths that lead to i . We will use the superscript to denote the number
 691 of steps for message passing.
 692

693 Let's consider the first message passing round, the original node i will receive messages from the slow nodes connected to
 694 it:

$$\begin{aligned} 695 \quad 696 \quad 697 \quad 698 \quad 699 \quad 700 \quad 701 \quad 702 \quad 703 \quad 704 \quad 705 \quad 706 \\ \forall i, x_i^{(1)} &= \sum_{(j,k) \in \mathcal{V}_i} a_{ij} \tilde{x}_k^{(0)} \\ &= \sum_{(j,k) \in \mathcal{V}_i} a_{ij} \left((1-\alpha)x_i^{(0)} + \alpha x_j^{(0)} \right) \\ &= (1-\alpha)x_i^{(0)} \left(\sum_{(j,k) \in \mathcal{V}_i} a_{ij} \right) + \alpha \cdot \sum_{(j,k) \in \mathcal{V}_i} a_{ij} x_j^{(0)} \\ &= (1-\alpha)x_i^{(0)} + \alpha \cdot \sum_{(j,k) \in \mathcal{V}_i} a_{ij} x_j^{(0)} \end{aligned}$$

707 x_i denotes an i.i.d. sample from the underlying latent generative model. Now if we use Lemma 1 in (Keriven, 2022) and
 708 leverage our assumption on the Gaussianity of our features and a large number of nodes, it holds with high probability, that
 709 the features after one round of message passing can be approximated as:

$$710 \quad 711 \quad 712 \quad 713 \quad 714 \quad \forall i, x_i^{(1)} \simeq (1-\alpha)x_i + \alpha Ax_i \tag{4}$$

Note that in the original analysis, they prove that $x^{(1)} \simeq Ax^{(0)}$. Our model, however, scales the messages from the rest of
 the network by α while the self-information is preserved and scaled by $1-\alpha$.

715 The slow nodes are also updated, since they have a single incoming edge, they simply copy the feature of their corresponding
 716 source node:

$$717 \quad 718 \quad 719 \quad \forall i, \forall (j, k) \in \mathcal{V}_i, \quad \tilde{x}_k^{(1)} = x_j^{(0)} \quad (5)$$

720 Let's perform a second round of message passing, we apply the same process again. The original nodes are updated as
 721 follows:

$$722 \quad 723 \quad 724 \quad 725 \quad 726 \quad 727 \quad 728 \quad \begin{aligned} \forall i, x_i^{(2)} &= \sum_{(j,k) \in \mathcal{V}_i} a_{ij} \tilde{x}_k^{(1)} \\ &= \sum_{(j,k) \in \mathcal{V}_i} a_{ij} x_j^{(0)} \\ &\simeq Ax_i^{(0)} \end{aligned}$$

729 We note that after the second round of message passing, the feature embeddings of the original nodes, are identical to the
 730 feature embeddings we would have obtained without HalfHop after a single step of message passing.

731 The slow nodes receive a copy of the embedding of their source node:

$$732 \quad 733 \quad 734 \quad 735 \quad \forall i, \forall (j, k) \in \mathcal{V}_i, \quad \tilde{x}_k^{(2)} = x_j^{(1)} \simeq (1 - \alpha)x_j + \alpha Ax_j$$

736 We can write a general formula where at each step, the original nodes receive and aggregate messages from the slow nodes
 737 (we note this operation AGG). The slow nodes are updated based on the source node they are connected to. We write this as:

$$738 \quad 739 \quad 740 \quad x_i^{(t+1)} = \text{AGG}(\tilde{x}_k^{(t)}), \quad \tilde{x}_k^{(t+1)} = x_j^{(t)}. \quad (6)$$

741 We can now apply the recursive formulas (6) iteratively. This assumes that we can approximate the AGG operation with
 742 a multiplication of A for Half-Hop. This is due to the underlying model assumptions which imply that the node feature
 743 distribution remains the same after a linear combination of i.i.d. Gaussian variables and the node features are approximately
 744 independent when n is large. Hence, if we continue applying the recursive formula and replace each AGG operation over x
 745 by Ax , then for any $k \geq 0$, mathematical induction yield

$$746 \quad 747 \quad x_i^{(2t+1)} = \alpha A^{(t+1)} x_i + (1 - \alpha) A^{(t)} x_i, \quad \tilde{x}_k^{(2t+1)} = A^t x_j.$$

748 Now we complete the proof by recalling that the covariance of the original node feature x is Σ and hence, the updated
 749 covariance after k steps of message passing is:

$$750 \quad 751 \quad 752 \quad \Sigma' = \frac{1}{2} A^{k-1} \left(\text{Id} + ((1 - \alpha)\text{Id} + \alpha A)^2 \right) \Sigma. \quad (7)$$

753 \square End of proof.

754 **Remark.** The modified covariance term in Equation 7 consists of two main terms, the first being a component that smooths
 755 the original covariance with A at a rate of $k - 1$, which is roughly half the original rate of smoothing for the graph without
 756 Half-Hop. In addition to this first slower smoothing term, we also find a second contribution to the new covariance. The
 757 second modified smoothing term is $((1 - \alpha)\text{Id} + \alpha A)^2$, where in this case we see a uniform boosting of the covariance
 758 spectrum coming from the first term and weighted by $(1-\alpha)$, and a second term coming from a rescaling of A . Thus, for
 759 small values of α we can interpret this as having a strong boosting of the self-loops in the graph. We also confirm that this is
 760 indeed the case for our analysis of the receptive field for different values of α in Figure 3.

761 B. Ablations

762 B.1. Testing the directionality of edges added in Half-Hop

763 Recall that for an edge e_{ij} , Half-Hop introduces a new slow node ν along the edge from v_i to v_j as follows:

$$764 \quad 765 \quad \text{HH : } v_i \rightarrow \nu \leftrightarrow v_j.$$

770 In this experiment, we try alternative connectivity schemes. In the first variant, $\text{HH}^{(1)}$, we do not introduce a backward edge
 771 that goes from the destination to the slow node:

$$772 \quad 773 \quad \text{HH}^{(1)} : v_i \rightarrow \nu \rightarrow v_j.$$

775 The second variant, $\text{HH}^{(2)}$, we add an edge going from the slow node to the source node:

$$777 \quad 778 \quad \text{HH}^{(2)} : v_i \leftrightarrow \nu \leftrightarrow v_j.$$

780 We assess the influence of the above connectivity schemes for the vanilla GCN model across three different datasets - Texas,
 781 Actor and Cornell, and we tabulate our results in Table 1. Significantly better performance can be noted for our proposed
 782 connectivity scheme HH, which also happens to be the more intuitive solution since it respects the directionality of the
 783 original edge, and the slow node gets to communicate with both source and target.

784 The training scheme, hyperparameters, and the model used for this experiment are the same as that of the results reported in
 785 the paper for heterophilic datasets.
 786

Dataset	HH $v_i \rightarrow \nu \leftrightarrow v_j$	$\text{HH}^{(1)}$ $v_i \rightarrow \nu \rightarrow v_j$	$\text{HH}^{(2)}$ $v_i \leftrightarrow \nu \leftrightarrow v_j$
Texas	71.71 ± 8.76	68.8 ± 6.50	58.47 ± 5.56
Actor	33.35 ± 1.00	32.17 ± 0.84	31.93 ± 1.26
Cornell	63.42 ± 5.62	57.66 ± 6.89	42.16 ± 6.57

792 *Table 1. Ablations of different connectivity motifs for slow nodes on heterophilic datasets.* The three connectivity motifs are displayed
 793 along the columns, from left to right, Half-Hop as proposed, $\text{HH}^{(1)}$ which still introduces delay but doesn't add a backward edge from
 794 the target node to slow node, and $\text{HH}^{(2)}$ which doesn't have two bi-directional edges and hence doesn't have the same delayed message
 795 passing effect. The performance values correspond to that of vanilla GCN model coupled with the respective HH variant.

798 B.2. Testing different slow node initializations

800 In this experiment, we ablate the linear interpolation initialization we use for the slow node (Section 2.2). We test two
 801 simpler alternatives: 1) 'Zero': Setting the initial features to zero, 2) 'Random': Initializing the with random values in the
 802 range of [0,1].

803 As tabulated in Table 2, from the drop in performance as we move from 'Weighted Sum' to 'Zero' to 'Random' we can
 804 clearly observe a significant gain upon using our proposed 'Weighted Sum' scheme. This would be as the alpha parameter
 805 in the weighted sum scheme enables the new slow node to be initialized with a blend of information from the source and
 806 target node embeddings.

807 For this experiment too we use the same hyperparameters, training scheme, and model as used for the results reported in the
 808 paper for heterophilic datasets.
 809

Dataset	Weighted Sum	Zeros	Random
Texas	72.88 ± 7.17	61.8 ± 5.91	53.33 ± 5.27
Actor	33.39 ± 1.29	28.93 ± 2.83	24.7 ± 1.16
Cornell	63.33 ± 5.70	49.55 ± 7.06	37.48 ± 6.93

815 *Table 2. Ablation of weighted sum method of initializing slow node embeddings* The performance values correspond to that of vanilla
 816 GCN model coupled with HH utilizing respective node embedding initialization

819 C. Details for Supervised Experiments

820 For comprehensive analysis on the full-spectrum of real-world graphs, we study the effects of our approach on both
 821 homophilic and heterophilic datasets.

823 **Homophily.** We follow the definition of *node homophily ratio* as used in (Pei et al., 2020) given by the formula:
 824

$$\frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{(v, w) : w \in \mathcal{N}(v) \wedge y_v = y_w\}|}{|\mathcal{N}(v)|}$$

In the above formula - \mathcal{V} denotes the set of all nodes in the graph, $\mathcal{N}(v)$ denotes all the neighbors of an arbitrary node v , and y_v denotes the class membership of the node $v \in \mathcal{V}$.

In our experiments, we treat datasets with Homophily Level ≥ 0.5 as *Homophilic datasets* and rest as *Heterophilic datasets*.

C.1. Homophilic Datasets

Amazon Computers and Amazon Photos (McAuley et al., 2015): Graphs pertaining to Amazon’s co-purchasing data. Products (nodes) are linked with each other if they are frequently bought together. Each node’s features are the product’s word embeddings (Bag of words). The task is to classify the nodes into product categories.

Coauthor CS and Coauthor Physics (Sinha et al., 2015): Nodes represent word embeddings (Bag of words) of particular keywords used by authors in their articles, and edges indicate collaboration between authors. The task is to categorize the authors into particular research fields based on their keywords and collaborations.

WikiCS (Mernyei & Cangea, 2020): The network graph of Computer Science Wikipedia articles. Each node is an article with features as word embeddings (GLoVe) of the article content and the edge represents the link. The task is to classify the node to one of 10 article categories.

Basic statistics of the homophilic datasets used are listed in Table 3. The experimental setup follows that of (Luo et al., 2022), where we split the dataset into development and test sets. All the hyperparameter tuning is done on the development set and the best models are evaluated on the test set. The runs are averaged over 20 random splits to minimize noise. We follow a 60:20:20% train/val/test split for the Amazon and Coauthor datasets, and 20 pre-split masks provided in the WikiCS dataset.

Table 3. Statistics of homophilic datasets used in our experiments

	Nodes	Edges	Features	Classes	Node Homophily Ratio
Amazon Photos	7,650	119,081	745	8	0.8365
Amazon Computers	13,752	245,861	767	10	0.7853
Coauthor CS	18,333	81,894	6,805	15	0.8320
Coauthor Physics	34,493	247,962	8,415	5	0.9153
Wiki CS	11,701	216,123	300	10	0.6588

C.2. Heterophilic Datasets

Texas, Wisconsin, Actor, Chameleon & Cornell (Luan et al., 2022): five real-world datasets with graphs that have a homophily level ≤ 0.30 . Basic statistics of the datasets are listed in Table 4.

Table 4. Statistics of heterophilic datasets used in our experiments

	Nodes	Edges	Classes	Node Homophily Ratio
Texas	183	295	5	0.11
Wisconsin	251	488	5	0.21
Film	7,600	26,752	5	0.22
Squirrel	5,201	198,493	5	0.22
Chameleon	2,277	31,421	5	0.23
Cornell	183	280	5	0.30

We full the experimental setup in (Pei et al., 2020), we use the same 10 train/val/test splits that are provided. We also perform similar hyperparameter tuning using randomized grid search using only the train and validation sets. Once we find the best model, we report the accuracy on the test set, which is only seen once. We include the final hyperparameters of the best models for each architecture and dataset in Table 5.

880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934

Table 5. Heterophilic Datasets.

Dataset	Model	lr	weight decay	depth	hidden	dropout	α	p
Texas	HH-GCN	0.0291	0.0096	2	64	0.8058	0.0043	0.9526
	HH-GraphSAGE	0.0170	0.0053	2	64	0.1967	0.9397	0.7140
	HH-GAT	0.0328	0.0066	2	32	0.1288	0.0902	0.9841
Wisconsin	HH-GCN	0.0105	0.0002	3	128	0.6612	0.9937	0.7140
	HH-GraphSAGE	0.0202	0.0042	3	64	0.3462	0.0100	0.6177
	HH-GAT	0.0539	0.0068	3	16	0.2141	0.0026	0.9797
Actor	HH-GCN	0.0313	0.0087	3	64	0.5511	0.0369	0.5466
	HH-GraphSAGE	0.0133	0.0090	3	32	0.3737	0.0116	0.8368
	HH-GAT	0.0009	0.0001	3	128	0.8708	0.0549	0.9594
Squirrel	HH-GCN	0.0053	0.0001	3	128	0.2455	0.0145	0.8257
	HH-GraphSAGE	0.0296	0.0001	2	128	0.8668	0.9474	0.5198
	HH-GAT	0.0027	0.0001	3	64	0.5131	0.9277	0.1549
Chameleon	HH-GCN	0.0318	0.0057	2	128	0.8040	0.0510	0.9986
	HH-GraphSAGE	0.0225	0.0001	2	32	0.7175	0.9834	0.6226
	HH-GAT	0.0012	0.0008	3	64	0.0439	0.9766	0.9386
Cornell	HH-GCN	0.0505	0.0055	2	32	0.4123	0.0145	0.9660
	HH-GraphSAGE	0.0697	0.0018	2	64	0.0697	0.8807	0.5660
	HH-GAT	0.0572	0.0070	2	64	0.0572	0.0710	0.9979

Table 6. Hyperparameter settings for unsupervised BGRL learning with Half-Hop

Aug. Hyperparameters	Am. Comp.	Am. Photos	Co. CS	Co. Phy	WikiCS
$p_{hh,1}$	0.75	0.75	0.75	0.75	0.75
$p_{hh,2}$	0.75	0.75	0.75	0.75	0.75
α_1	0.50	0.50	0.50	0.50	0.50
α_2	0.50	0.50	0.50	0.50	0.50
$p_{f,1}$	0.20	0.10	0.30	0.10	0.20
$p_{f,2}$	0.10	0.20	0.40	0.40	0.10
$p_{e,1}$	0.50	0.40	0.30	0.40	0.20
$p_{e,2}$	0.40	0.10	0.20	0.10	0.30

D. Details for Self-supervised Experiments

We use the same real-world datasets (Amazon-Photos, Amazon-Computers, Coauthor-CS and Coauthor-Physics) as the ones used in supervised setting. The whole dataset graph is used to train the self-supervised learning model as a transductive task, and for the linear evaluation phase, the dataset is split into train/val/test in the (10:10:80%) format respectively following the same setup as (Thakoor et al., 2021).

When using GRACE (Zhu et al., 2020b) and BGRL (Thakoor et al., 2021), we refer to the hyperparamters reported in their respective works. The edge-dropout and feature-dropout hyperparameters used are *edge-masking probability* for both the views ($p_{e,1}, p_{e,2}$), and *feature-masking probability* for both the views ($p_{f,1}, p_{f,2}$). The Half-Hop hyperparameters used for the training are *Half-Hop probability* for both the views ($p_{hh,1}, p_{hh,2}$) and the coefficient used for combining features for both the views (α_1, α_2). Details of their values are tabulated in Table 6.

The method of evaluation follows the linear evaluation protocol (Veličković et al., 2019), where the learned weights of the self-supervised learning model are frozen and the output representations are used to tune and train a Linear Classifier (without propagating gradients to the encoder). We use an l_2 -regularized Logistic Regression Model with a *liblinear* solver from the Scikit-learn library (Pedregosa et al., 2011).

E. Comparisons with other GNNs

There have been multiple modifications on top of traditional GNN architectures to optimize for the task of heterophilic node classification. In Table 7 we detail the different architectural components, losses, and pieces of information that are used to improve performance for heterophilic datasets. In the table, we breakdown the different components of popular methods that we compare with in the main text, including: MixHop (Abu-El-Haija et al., 2019), (ii) GGCN (Yan et al., 2021), and (iii) H_2 GCN (Zhu et al., 2020a).

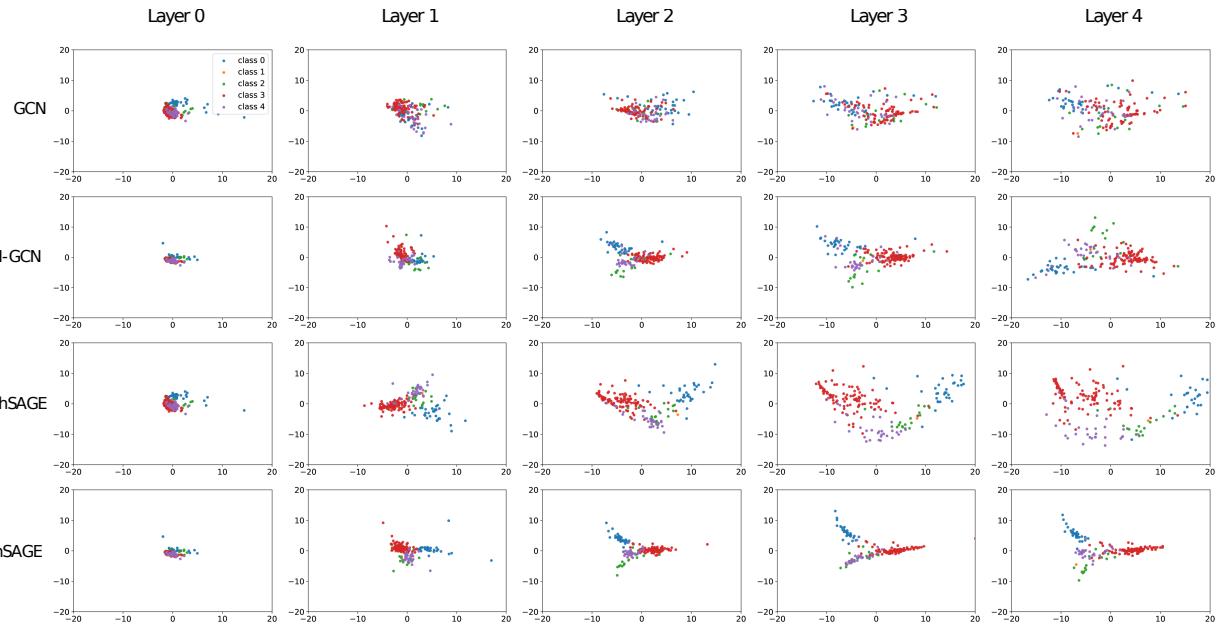
	Higher-order neighbors	Weights for self-loops	Concat across layers	Dynamic gating
GCN	✗	✗	✗	✗
SAGE	✗	✓	✗	✗
MixHop	✓	✗	✓	✗
GGCN	✗	✗	✗	✓
H_2 GCN	✓	✓	✓	✗
HH-GCN	✗	✗	✗	✗
HH-SAGE	✗	✓	✗	✗

Table 7. Different components used in graph neural networks optimized for heterophilic node classification. From left to right, we show methods that incorporate additional information from higher-order neighbors, separate weights for self-loops, and other additional components. Here, we observe the fact that Half-Hop is lightweight and doesn't require extra components in the loss and also doesn't explicitly compute separate weights for self-loops.

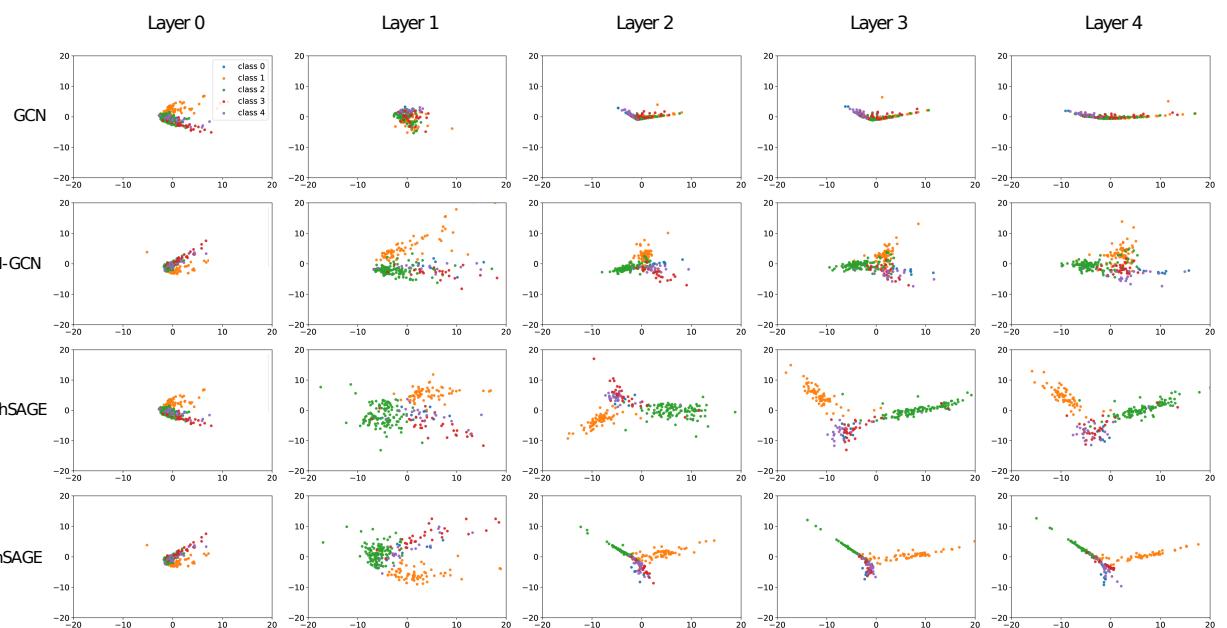
F. Additional Experiments**F.1. Latent Visualization across GNN Layers**

In this experiment, we visualize the latent space of node embeddings across various depths, with and without Half-Hop (Figures 1, 2 and 3) using a GCN and GraphSAGE encoder.

Observing the latent space visualizations, we can make a few interesting observations. Upon applying Half-Hop, the embeddings of the same class appear to aggregate together better while the embeddings of different classes seem to maximally distance themselves from each other and from the center of the latent space. Though this is slightly observed in all cases, this can be clearly noted in the case of GraphSAGE + Half-Hop for the Citeseer dataset in Figure 3. Visualizations of latents for vanilla GCN (without Half-Hop) indicate poor class separation (Figure 1 and Figure 2) and this can be seen reflected directly in vanilla GCN's performance as noted in Table 2 in the paper. On a more general note, class separation appears best after the second and third layers as observed across Figures 1, 2 and 3. This is similar to what is observed in terms of raw performance - the best-performing models are 2 or 3 layers deep.

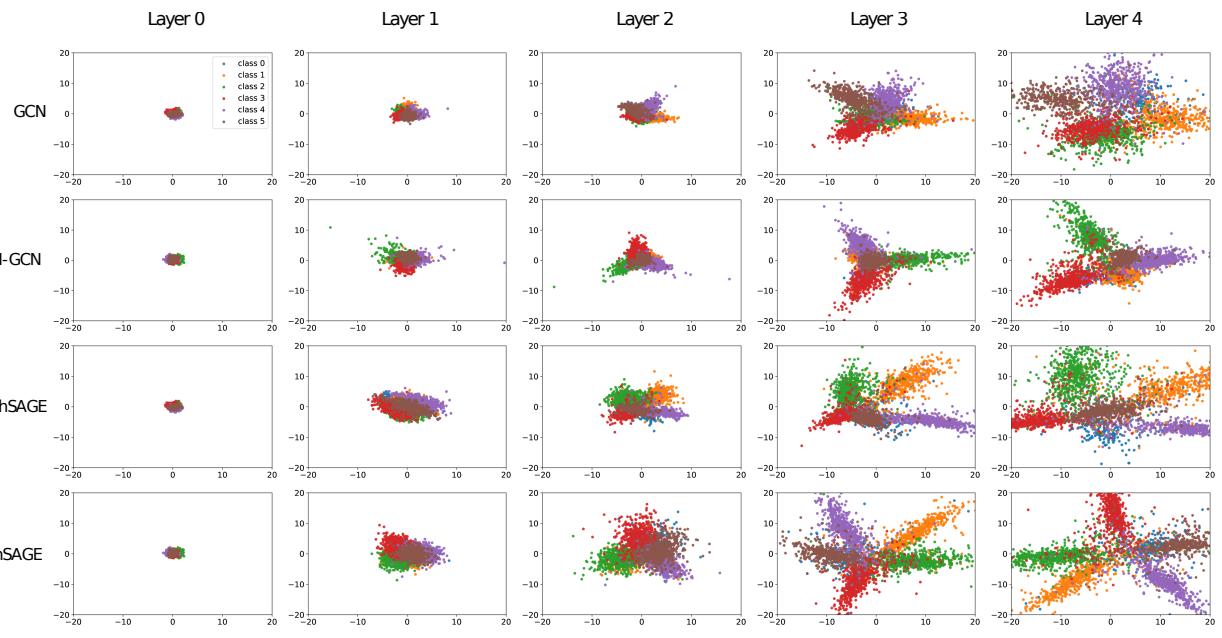


1012 Figure 1. Latent space visualizations with and without Half-Hop for Texas dataset. We can note how similar data points cluster closer
1013 together in the cases where Half-Hop is applied. As class 1 is very rare, it is not clearly visible in the visualization. 'Layer 0' denotes the
1014 latent space before the GNN layers are applied.



1040 Figure 2. Latent space visualizations with and without Half-Hop for Wisconsin dataset. Though the color scheme is the same, it appears
1041 different from that of Figure 1 due to the difference in the distribution of nodes across classes in Texas and Wisconsin datasets.

1042
1043
1044



1081 *Figure 3.* Latent space visualizations with and without Half-Hop for Citeseer dataset. We can again note how in the cases where Half-Hop
1082 is applied, similar latents seem to cluster closer together. Citeseer has a larger number of nodes compared to Texas and Wisconsin datasets
1083 and hence the latents in these visualizations appear to be packed more densely in comparison.

1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099