

## APPENDIX

Table 5: Summary of key notation.

Symbol	Description
$D$	set of nodes for training or inference
$D^{(i)}$	$i^{th}$ bootstrap of relatives sampled from $D$
$D_u^{(i)}$	relative of node $u$ sampled for the $i^{th}$ bootstrap
$\mathcal{E}_k$	set of edges in $\mathcal{G}_k$
$\mathcal{G}_k$	graph with max order $k$ ( $k = 1$ is a FON, $k > 1$ is a HON)
$\text{GNN}_i(\cdot)$	the $i^{th}$ learner (GNN) in the ensemble
$\mathbf{h}_u^{(i,t)}$	hidden representation of node $u$ at $t^{th}$ layer (timestamp) of learner $i$
$\ell$	number of base learners in the ensemble
$\mathcal{N}_k(u)$	neighborhood of node $u$ in $\mathcal{G}_k$
$P_u^k$	distribution for sampling relatives from $\Omega_u^k$
$\mathcal{S}$	set of observed paths used to build $\mathcal{G}_k$
$\mathcal{V}_k$	set of nodes in $\mathcal{G}_k$
$w_k(u, v)$	weight of directed edge $(u, v)$ in $\mathcal{G}_k$
$\hat{\mathbf{y}}_u$	final (pooled) predicted label for node $u$
$\hat{\mathbf{y}}_u^{(i)}$	the $i^{th}$ learner’s predicted label for node $u$
$\Omega_u^k$	relatives of node $u$ in $\mathcal{G}_k$

## A PROOF OF THEOREM 1

**Preliminaries** Without loss of generality, in this section we treat the neighborhood  $\mathcal{N}_k(u')$  in a graph  $\mathcal{G}_k$  for  $k \geq 1$  as only the out-neighbors of  $u'$  (i.e.  $\mathcal{N}_k(u') = \{v \in \mathcal{V}_k, w_k(u', v) > 0\}$ ) and ignore in-edges. Given a node  $u' \in \mathcal{V}_k$ , we first define the probability that a random walker over  $\mathcal{G}_k$  would move from  $u'$  to any  $v \in \mathcal{N}_k(u')$ :

$$\pi_k(u' \rightarrow v) = \frac{w_k(u', v)}{\text{outdeg}_k(u')}. \quad (6)$$

Given a FON  $\mathcal{G}_1$  and any  $u \in \mathcal{V}_1$ , let  $u' \in \Omega_u^k$  be any relative of  $u$  in a HON  $\mathcal{G}_k$  that is constructed from the same data as  $\mathcal{G}_1$ . Following Saebi et al. (2020d), we next define the Kullback-Leibler divergence of  $\mathcal{N}_k(u')$  with respect to  $\mathcal{N}_1(u)$ :

$$D_{KL}(\mathcal{N}_k(u') \parallel \mathcal{N}_1(u)) = \sum_{v \in \mathcal{N}_1(u)} \pi_1(u \rightarrow v) \log_2 \frac{\pi_1(u \rightarrow v)}{\pi_k(u' \rightarrow v)}. \quad (7)$$

Our notation throughout this section assumes that  $v \in \mathcal{N}_1(u)$  and  $v \in \mathcal{N}_k(u')$ ; however, a relative  $v' \in \Omega_u^k$  often replaces  $v$  in  $\mathcal{N}_k(u')$ . For example, in Figure 1, C|A is substituted for its base node C in the neighborhood of A, but the edge (A, C|A) still fundamentally represents a step from C to A. GrowHON (Krieg et al., 2020a) constructs  $\mathcal{G}_k$  such that  $u' \in \mathcal{V}_k$  iff the following inequality holds:

$$D_{KL}(\mathcal{N}_k(u') \parallel \mathcal{N}_1(u)) > \frac{m}{\log_2(1 + \text{freq}(u'))}, \quad (8)$$

where  $m > 1$  is length of the sequence encoded by  $u'$  (see Section 2) and  $\text{freq}(u') \geq 1$  is the number of times  $u'$  is found in all the observed paths in  $\mathcal{S}$ .

**Lemma 1.** *If  $u \in \mathcal{V}_1$  and  $u' \in \Omega_u^k$ , then there exists at least one node  $v \in \mathcal{N}_1(u)$  such that  $\pi_1(u \rightarrow v) \neq \pi_k(u' \rightarrow v)$ .*

*Proof.* If  $u' \in \Omega_u^k$ , then  $D_{KL}(\mathcal{N}_k(u') \parallel \mathcal{N}_1(u)) > 0$ . This is because  $\text{freq}(u') \geq 1$ , so  $\log_2(1 + \text{freq}(u')) \geq 1$  and the right side of Eq. 8 cannot be negative. If  $D_{KL}(\mathcal{N}_k(u') \parallel \mathcal{N}_1(u)) = 0$ , then the inequality in Eq. 8 cannot hold, so  $u' \notin \mathcal{V}_k$  and, consequently,  $u' \notin \Omega_u^k$ . Additionally, if  $\pi_1(u \rightarrow v) = \pi_k(u' \rightarrow v)$  for all  $v \in \mathcal{N}_1(u)$ , then by Gibbs' inequality we have  $D_{KL}(\mathcal{N}_k(u') \parallel \mathcal{N}_1(u)) = 0$ . Therefore, since  $u' \in \Omega_u^k$ , there must exist at least one node  $v \in \mathcal{N}_1(u)$  such that  $\pi_1(u \rightarrow v) \neq \pi_k(u' \rightarrow v)$ .  $\square$

We now restate and prove Theorem 1.

**Theorem 1.** *Let  $\mathcal{G}_1$  and  $\mathcal{G}_k$  be a FON and HON, respectively, both constructed from the same input  $\mathcal{S}$ . Let  $\mathcal{N}_1(u)$  and  $\mathcal{N}_k(u')$  denote the neighborhoods of any node  $u$  in  $\mathcal{G}_1$  and  $\mathcal{G}_k$ , respectively. Let  $\text{AGGREGATE}(\cdot)$  represent any symmetric neighborhood aggregation function. If  $u \in \mathcal{V}_1$  and  $u' \in \Omega_u^k$ , then  $\text{AGGREGATE}(\mathcal{N}_k(u'))$  is a biased estimator of  $\text{AGGREGATE}(\mathcal{N}_1(u))$ .*

*Proof.* Without loss of generality, we consider the case in which AGGREGATE operates on a sample of neighbors (i.e., GraphSAGE (Hamilton et al., 2017)). Let  $V \sim \mathcal{N}_k(u')$  denote a random sample drawn from  $\mathcal{N}_k(u')$ . Assuming we weight the sampling distribution according to edge weights, we can write the probability of sampling any node  $v$  as

$$p(V = v) = \frac{w_k(u', v)}{\text{outdeg}_k(u')} = \pi_k(u' \rightarrow v), \quad (9)$$

where  $w_k(u', v)$  is the weight of edge  $(u', v)$  in  $\mathcal{G}_k$  (Section 2) and  $\text{outdeg}_k(u')$  is the weighted out-degree of  $u'$  in  $\mathcal{G}_k$ . If we say that each  $u \in \mathcal{V}_k$  is represented by a real-valued feature vector  $\mathbf{x}_u^k = [x_{u,1}^{(k)}, x_{u,2}^{(k)}, \dots, x_{u,d}^{(k)}]$ , then, for a single sample drawn from  $\mathcal{N}_k(u')$ , we can formulate the expected value for each vector as follows:

$$\mathbb{E}_{V \sim \mathcal{N}_k(u')} [\mathbf{x}_V^k] = p(V = v) \mathbf{x}_v^k. \quad (10)$$

If we combine these feature vectors for each of the  $n$  nodes in  $\mathcal{V}_k$ , then we can represent the full expectation of  $V$  as a neighborhood matrix

$$\begin{aligned}\mathbb{E}_{V \sim \mathcal{N}_k(u')} [V] &= \begin{bmatrix} p(V=1) \mathbf{x}_1^k \\ p(V=2) \mathbf{x}_2^k \\ \vdots \\ p(V=n) \mathbf{x}_n^k \end{bmatrix} \\ &= \begin{bmatrix} p(V=1) x_{1,1}^{(k)} & p(V=1) x_{1,2}^{(k)} & \dots & p(V=1) x_{1,d}^{(k)} \\ p(V=2) x_{2,1}^{(k)} & p(V=2) x_{2,2}^{(k)} & \dots & p(V=2) x_{2,d}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ p(V=n) x_{n,1}^{(k)} & p(V=n) x_{n,2}^{(k)} & \dots & p(V=n) x_{n,d}^{(k)} \end{bmatrix}.\end{aligned}$$

We can simplify our notation by assuming the initial feature vectors are identity features (i.e., one-hot encodings of the node indices) defined on the base nodes, i.e., for features of nodes in  $\mathcal{V}_1$ , we have  $x_{u,j}^{(1)} = \delta_{uj}$  for all  $u, j \leq d = n$ , where  $\delta$  is the Kronecker delta. For the features of nodes in  $\mathcal{V}_k$ , we assume that each  $u' \in \Omega_u^k$  uses the same features as its base node, i.e.,  $x_{u',j}^{(k)} = \delta_{uj}$ . Substituting these values in the above matrix gives

$$\mathbb{E}_{V \sim \mathcal{N}_k(u')} [V] = \begin{bmatrix} p(V=1) & 0 & \dots & 0 \\ 0 & p(V=2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & p(V=n) \end{bmatrix}.$$

Without loss of generality, let AGGREGATE be the feature-wise MEAN of  $N$  samples drawn from  $\mathcal{N}_k(u')$ . We can then represent the expectation of AGGREGATE as

$$\begin{aligned}\mathbb{E}_{V \sim \mathcal{N}_k(u')} [\text{AGGREGATE}(V)] &= \begin{bmatrix} \frac{1}{N} \\ \frac{1}{N} \\ \vdots \\ \frac{1}{N} \end{bmatrix} \begin{bmatrix} Np(V=1) & 0 & \dots & 0 \\ 0 & Np(V=2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Np(V=n) \end{bmatrix} \\ &= \begin{bmatrix} p(V=1) \\ p(V=2) \\ \vdots \\ p(V=n) \end{bmatrix} \\ &= \begin{bmatrix} \pi_k(u' \rightarrow 1) \\ \pi_k(u' \rightarrow 2) \\ \vdots \\ \pi_k(u' \rightarrow n) \end{bmatrix}. \quad (\text{via Eq. 9})\end{aligned}$$

If  $V \sim \mathcal{N}_1(u)$ , we instead have

$$\mathbb{E}_{V \sim \mathcal{N}_1(u)} [\text{AGGREGATE}(V)] = \begin{bmatrix} \pi_1(u \rightarrow 1) \\ \pi_1(u \rightarrow 2) \\ \vdots \\ \pi_1(u \rightarrow n) \end{bmatrix}.$$

From Lemma 1 we know that there exists at least one  $v \in \mathcal{N}_1(u)$  such that  $\pi_1(u \rightarrow v) \neq \pi_k(u' \rightarrow v)$ . Therefore,  $\mathbb{E}_{V \sim \mathcal{N}_k(u')} [\text{AGGREGATE}(V)] \neq \mathbb{E}_{V \sim \mathcal{N}_1(u)} [\text{AGGREGATE}(V)]$ , and  $\text{AGGREGATE}(\mathcal{N}_k(u'))$  is a biased estimator of  $\text{AGGREGATE}(\mathcal{N}_1(u))$ .  $\square$

This result holds for other common aggregators like SUM and MAX for single samples. For MAX, however, the sample size matters, because the expected value for the  $j^{th}$  feature is the probability that  $j$  is sampled at least once. We conjecture that this is problematic for larger sample sizes in weighted graphs, and is perhaps the reason why mean-pooling outperformed max-pooling for our GraphSAGE baseline during our initial experiments (Section C).

## B DATA DETAILS AND EXPERIMENTAL SETUP

**Air:** Flight trajectories of passenger itineraries in the United States. Nodes represent airport locations (cities) and edges represent passengers flying between locations. The itineraries were retrieved from the Airline Origin and Destination Survey (DB1B) database, which is publicly available through the U.S. Bureau of Transportation Statistics<sup>3</sup>. We downloaded all records between Jan. 1 and Dec. 31, 2019 from the DB1BCoupon table, joined and sorted itineraries using the “ITIN\_ID” and “SEQ\_NUM” columns, and discarded any itineraries with missing origin or destination information. Node classes represent the geographical location of each airport, aggregated by standard federal region of the United States. While we extracted this particular set of paths, this database has been utilized in prior work on HONs (Rosvall et al., 2014; Scholtes, 2017).

**T2D:** Disease trajectories for type 2 diabetes patients in the state of Indiana (Krieg et al., 2020b). Nodes represent ICD9 diagnosis codes and edges represent sequential diagnoses. Following prior work, we only preserved the first occurrence of each diagnosis code for each patient; however, we did not split trajectories based on the period of time between diagnoses. Node classes are the chapters of each ICD9 code, which represent categories of disease classification.

**Wiki:** Clickstreams of users playing the Wikispeedia game, in which a player attempts to navigate from a source to a target article by clicking only Wikipedia links (West et al., 2009). Here, nodes represent Wikipedia articles, and edges represent user clicks between articles. We included both finished and unfinished paths. Node classes are the subjects of each article.

**Mag:** Readership trajectories for a large online magazine from Jan 1. to Apr. 15, 2020 (Wang et al., 2020). Nodes represent online content (articles, games, etc.), and edges represent sequential clicks by users. We only considered trajectories in which the user visited at least three nodes during a session. Node classes represent the content type (magazine, culture, news, or humor).

**Mag+:** An expanded version of Mag which also includes data from July 1, 2019 through Dec. 31, 2019, as well as two additional content types (books, home). Due to its size and similarity to Mag, we excluded it from link prediction experiments.

**Ship:** Trajectories of global shipping activity (Saebi et al., 2020c). Nodes represent ports, and edges represent ships traveling between ports. Node classes are provinces, as defined by the Marine Ecoregions of the World (MEOW) system for classifying oceans and waterways (Spalding et al., 2007).

For node classification, we used stratified 5-fold cross validation (Shchur et al., 2018) and reported the mean micro F1-score. For link prediction, we generated training and testing sets by randomly sampling 10% of positive node pairs and an equal number of negative node pairs. In addition to hiding all edges (in both directions) between testing pairs, for experiments using  $\mathcal{G}_2$  we also hid all edges between any pair of nodes in the same higher-order families as each testing pair. We repeated each experiment five times and reported the mean area under the precision-recall curve (AUPRC) (Yang et al., 2015). All experiments utilized the same training and testing splits. For models trained on  $\mathcal{G}_1$  we used identity features (i.e., one-hot encodings of the node indices) for each node. For models trained on  $\mathcal{G}_2$ , we used the same features as  $\mathcal{G}_1$  such that each  $u' \in \Omega_u^2$  shared the same features as its base node  $u$ .

<sup>3</sup><https://transtats.bts.gov/>. Accessed April 14, 2022.

## C MODEL TUNING AND HYPERPARAMETER SETUP

We manually tuned hyperparameters for each model. Table 6 summarizes the configurations that we evaluated in reporting the main results in Tables 2 and 3. We used the Adam optimizer with a learning rate of 0.0005 for GAT and GATv2, and a learning rate of 0.01 for all other models. For minibatch models (GraphSAGE, GIN, and DGE), we tested batch sizes of 16, 32, and 64. Unless otherwise noted, all results reported in Tables 2, 3, 7, and 8 were the best-performing configuration for each model as averaged across all testing folds.

In general, the most impactful hyperparameter was the number of GNN layers. As discussed in Section 4.2, on Air, T2D, and Mag, all baselines performed best with only a single layer for both node classification and link prediction. On Wiki, several models performed best with 3 layers (Figure 3). GCNII, which uses residual connections and is thus designed with deep GNNs in mind (Chen et al., 2020), was the only model we tested with more than 3 layers. For most models, increasing the number of hidden units per layer above 256 had little to no effect on performance. The exception to this was DGE-pool\* on Air, which performed best with 2,048 hidden units (Section 4.3).

Other hyperparameters varied by data set. The number of neighbors sampled ( $|\mathcal{N}|$ ) was very important on the dense graphs (T2D and Mag). In all cases, we found that increasing the sample size of the first layer was the most important, so we fixed the sample size at the second layer to 1 (per first-layer sample). Baselines preferred 512 samples for T2D and 256 for Mag. On the other data sets, 64 neighbor samples was sufficient for all models. DGE preferred 256 samples per base learner (the max that we tested) for T2D and 128 for Mag. The same trend held for GraphSAINT’s subgraph sampling, which preferred more roots on graphs that were dense and had more nodes (512 for Air and Wiki, 1024 for T2D, and 2048 for Mag, 4096 for Mag+, and 2048 for Ship). On all data sets, PathGCN performed best with at least 100 random walk samples per node.

For neighborhood aggregators, max-pooling was in all cases inferior to mean-pooling and sum-pooling, likely due to the inability of max-pooling to appropriately distinguish dense neighborhoods (Xu et al., 2019).

Table 6: Hyperparameters evaluated for each model.

Model-specific hyperparameters		
Model	# Layers	Other
GCN	{1, 2, 3}	—
GCNII	{1, 2, 3, 4, 6, 8}	$\alpha = 0.5, \lambda = 1.0$
GAT	{1, 2, 3}	attnheads = {4, 8, 16}
GATv2	{1, 2, 3}	attnheads = {4, 8, 16}
GraphSAGE	{1, 2, 3}	$ \mathcal{N}  = \{32, 64, 128, 256, 512, 1024\} \times \{1, 2, 4, 8\} \times \{1\}$ AGGREGATE = {MEANPOOL, MAXPOOL}
GIN	{1, 2, 3}	$ \mathcal{N}  = \{32, 64, 128, 256, 512, 1024\} \times \{1, 2, 4, 8\} \times \{1\}$
GraphSAINT	{1, 2, 3}	nroots = {256, 512, 1024, 2048, 4096}, walklen = {2, 3, 4}
PathGCN	{1, 2, 3}	nwalks = {10, 50, 100, 200}, walklen = {2, 3, 4, 5}
HO-GNN	{1, 2, 3}	—
SEAL	{1, 2, 3}	hops = {1, 2, 3}
DGE (all)	{1, 2, 3}	$\ell = \{4, 8, 16\},  \mathcal{N}  = \{32, 64, 128, 256\} \times \{1, 2, 4\} \times \{1\}$
General hyperparameters		
Model	Dropout	Hidden units (per layer)
All	0.4	{128, 256, 512, 1024, 2048}

## D ADDITIONAL POOLING MECHANISMS

In addition to Eqs. 5a, 5b, and 5c, we evaluated an attention mechanism (inspired by GAT (Veličković et al., 2018)) which used self-attention to pool the outputs of all sampled relatives. We computed attention coefficients  $e_{ij}$  for each  $i, j \leq \ell$  according to

$$e_{ij} = F\left(\mathbf{W}\left(\text{GNN}_i(D_u^{(i)})\right), \mathbf{W}\left(\text{GNN}_j(D_u^{(j)})\right)\right), \quad (11)$$

where  $F$  is a single-layer feed-forward network with a LeakyReLU activation,  $\mathbf{W} \in \mathbb{R}^{2d \times d}$  is a trainable weight matrix, and  $\text{GNN}_i(u) = \mathbf{h}_u^{(i,t)}$  (as in Eqs. 5a and 5b). We then computed the final attention coefficients  $\alpha_{ij}$  according to

$$\alpha_{ij} = \text{softmax}_i(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{m=1}^{\ell} \exp(e_{im})}. \quad (12)$$

We repeated this procedure for  $K$  separate attention heads. Whereas the original GAT layer averages the outputs of the attention heads before applying a non-linear activation (Veličković et al., 2018), we found that passing the same output to another feed-forward network provided better results. Formally, letting  $i$  be the index of we compute the vector of class probabilities  $\hat{\mathbf{y}}_u$  by

$$\hat{\mathbf{y}}_u = \sigma\left(F'\left(\sigma'\left(\frac{1}{K} \sum_{n=1}^K \left\| \sum_{j=1}^{\ell} \alpha_{ij}^{(n)} \mathbf{W}^{(n)} \text{GNN}_j(D_u^{(j)}) \right\| \right)\right)\right), \quad (13)$$

where  $\|$  is vector concatenation,  $F'$  is a feed-forward network with output dimension  $d/4$  (followed by batch normalization and a LeakyReLU activation),  $\alpha_{ij}^{(n)}$  and  $\mathbf{W}^{(n)}$  represent the attention coefficients and linear transformation, respectively, for the  $n^{\text{th}}$  attention head, and  $\sigma'$  is a LeakyReLU activation. For our experiments, we set  $K = 5$ ,  $d = 128$ , and a negative slope coefficient of 0.3 for all LeakyReLU activations. We refer to this model as DGE-attn and discuss its performance in Appendix E.

## E ADDITIONAL EXPERIMENTAL RESULTS

To support our claim that existing GNNs cannot effectively use  $\mathcal{G}_2$  as input, we also tested baselines using  $\mathcal{G}_2$  as the input graph for Air, T2D, Wiki, and Mag. As we expected, because each neighborhood in  $\mathcal{G}_2$  is only a subspace of its neighborhood in  $\mathcal{G}_1$ , performance was generally worse (or about the same) when compared to  $\mathcal{G}_1$ . Tables 7 and 8 detail the full results.

Table 7 also includes results for a GNN ensemble (DGE-bag) trained on  $\mathcal{G}_1$ , i.e. without relative sampling and pooling. Overall, we found that the performance was slightly better than GraphSAGE (the base GNN), but in all cases fell short of DGE-bag with relative sampling and pooling. The final result included in Table 7 is for one additional pooling mechanism, DGE-attn, which pooled representations from the sampled relatives via a self-attention mechanism. Since it did not generalize particularly well in the node classification experiments, we did not discuss it in the main manuscript and did not evaluate its performance on link prediction. More details are available in Appendix D. Table 8 also includes the full link prediction results for baselines, which we excluded from Table 3 because of space constraints in the main manuscript.

Figure 5 shows DGE-bag’s performance as a function of the number of base learners ( $\ell$ ) on four data sets. We found that 8-12 base learners produced consistently strong results. Figure 6 shows classifier accuracy and diversity (Section 4.3) for the T2D, Wiki, and Mag data sets, as a supplement to Figure 4.

Table 7: Additional node classification results (micro F1) for baselines using  $\mathcal{G}_2$  as input. Bold font indicates the best result for each data set.

Model	Input	Air	T2D	Wiki	Mag
GCN	$\mathcal{G}_1$	0.818 $\pm$ 0.03	0.480 $\pm$ 0.02	0.643 $\pm$ 0.01	0.796 $\pm$ 0.01
	$\mathcal{G}_2$	0.805 $\pm$ 0.05	0.500 $\pm$ 0.02	0.665 $\pm$ 0.02	0.744 $\pm$ 0.02
GCNII	$\mathcal{G}_1$	0.845 $\pm$ 0.05	0.511 $\pm$ 0.02	0.654 $\pm$ 0.02	0.801 $\pm$ 0.01
	$\mathcal{G}_2$	0.833 $\pm$ 0.05	0.537 $\pm$ 0.03	0.657 $\pm$ 0.02	0.816 $\pm$ 0.01
GAT	$\mathcal{G}_1$	0.804 $\pm$ 0.03	0.282 $\pm$ 0.10	0.639 $\pm$ 0.02	0.487 $\pm$ 0.06
	$\mathcal{G}_2$	0.817 $\pm$ 0.05	0.253 $\pm$ 0.03	0.646 $\pm$ 0.03	0.512 $\pm$ 0.04
GATv2	$\mathcal{G}_1$	0.838 $\pm$ 0.03	0.292 $\pm$ 0.07	0.643 $\pm$ 0.03	0.495 $\pm$ 0.05
	$\mathcal{G}_2$	0.842 $\pm$ 0.05	0.241 $\pm$ 0.10	0.646 $\pm$ 0.03	0.512 $\pm$ 0.13
GraphSAGE	$\mathcal{G}_1$	0.781 $\pm$ 0.04	0.654 $\pm$ 0.04	0.625 $\pm$ 0.02	0.808 $\pm$ 0.02
	$\mathcal{G}_2$	0.688 $\pm$ 0.02	0.516 $\pm$ 0.01	0.633 $\pm$ 0.01	0.818 $\pm$ 0.01
GIN	$\mathcal{G}_1$	0.745 $\pm$ 0.02	0.673 $\pm$ 0.04	0.636 $\pm$ 0.02	0.826 $\pm$ 0.02
	$\mathcal{G}_2$	0.660 $\pm$ 0.02	0.637 $\pm$ 0.03	0.625 $\pm$ 0.02	0.812 $\pm$ 0.02
GraphSAINT	$\mathcal{G}_1$	0.802 $\pm$ 0.02	0.600 $\pm$ 0.07	0.664 $\pm$ 0.01	0.821 $\pm$ 0.02
	$\mathcal{G}_2$	0.710 $\pm$ 0.03	0.567 $\pm$ 0.03	0.661 $\pm$ 0.02	0.828 $\pm$ 0.01
HONEM	$\mathcal{G}_1, \mathcal{G}_2$	0.805 $\pm$ 0.04	0.566 $\pm$ 0.02	0.588 $\pm$ 0.01	0.728 $\pm$ 0.02
DGE-bag	$\mathcal{G}_1$	0.772 $\pm$ 0.05	0.694 $\pm$ 0.04	0.645 $\pm$ 0.02	0.831 $\pm$ 0.01
DGE-concat	$\mathcal{G}_2$	0.825 $\pm$ 0.04	0.501 $\pm$ 0.06	0.615 $\pm$ 0.02	0.790 $\pm$ 0.02
DGE-concat*	$\mathcal{G}_2$	0.810 $\pm$ 0.04	0.439 $\pm$ 0.03	0.577 $\pm$ 0.02	0.761 $\pm$ 0.02
DGE-pool	$\mathcal{G}_2$	0.839 $\pm$ 0.03	0.735 $\pm$ 0.03	0.671 $\pm$ 0.01	0.860 $\pm$ 0.01
DGE-pool*	$\mathcal{G}_2$	<b>0.865</b> $\pm$ 0.02	0.555 $\pm$ 0.07	0.599 $\pm$ 0.04	0.775 $\pm$ 0.01
DGE-bag	$\mathcal{G}_2$	0.856 $\pm$ 0.02	<b>0.770</b> $\pm$ 0.04	<b>0.681</b> $\pm$ 0.00	<b>0.871</b> $\pm$ 0.01
DGE-bag*	$\mathcal{G}_2$	0.766 $\pm$ 0.04	0.719 $\pm$ 0.04	0.644 $\pm$ 0.02	0.841 $\pm$ 0.02
DGE-batch*	$\mathcal{G}_2$	0.764 $\pm$ 0.03	0.646 $\pm$ 0.01	0.623 $\pm$ 0.01	0.818 $\pm$ 0.01
DGE-attn	$\mathcal{G}_2$	0.829 $\pm$ 0.03	0.549 $\pm$ 0.05	0.618 $\pm$ 0.02	0.775 $\pm$ 0.05

\* shared parameters



Table 8: Full link prediction results (AUPRC) for baselines. Bold font indicates the best result for each data set.

Model	Input	Air	T2D	Wiki	Mag	Ship
GCN	$\mathcal{G}_1$	0.794 $\pm$ 0.01	0.793 $\pm$ 0.00	0.741 $\pm$ 0.01	0.742 $\pm$ 0.00	0.849 $\pm$ 0.00
	$\mathcal{G}_2$	0.777 $\pm$ 0.02	0.781 $\pm$ 0.00	0.749 $\pm$ 0.01	0.753 $\pm$ 0.00	—
GCNII	$\mathcal{G}_1$	0.806 $\pm$ 0.01	0.754 $\pm$ 0.01	0.781 $\pm$ 0.01	0.750 $\pm$ 0.00	0.848 $\pm$ 0.01
	$\mathcal{G}_2$	0.771 $\pm$ 0.01	0.732 $\pm$ 0.00	0.790 $\pm$ 0.01	0.759 $\pm$ 0.00	—
GAT	$\mathcal{G}_1$	0.786 $\pm$ 0.01	0.698 $\pm$ 0.02	0.794 $\pm$ 0.01	0.644 $\pm$ 0.00	0.764 $\pm$ 0.00
	$\mathcal{G}_2$	0.768 $\pm$ 0.01	0.681 $\pm$ 0.01	0.801 $\pm$ 0.01	0.641 $\pm$ 0.00	—
GATv2	$\mathcal{G}_1$	0.771 $\pm$ 0.01	0.701 $\pm$ 0.01	0.797 $\pm$ 0.01	0.645 $\pm$ 0.01	0.797 $\pm$ 0.00
	$\mathcal{G}_2$	0.784 $\pm$ 0.01	0.684 $\pm$ 0.00	0.796 $\pm$ 0.01	0.660 $\pm$ 0.01	—
GraphSAGE	$\mathcal{G}_1$	0.782 $\pm$ 0.02	0.713 $\pm$ 0.01	0.758 $\pm$ 0.01	0.717 $\pm$ 0.00	0.820 $\pm$ 0.00
	$\mathcal{G}_2$	0.757 $\pm$ 0.02	0.712 $\pm$ 0.01	0.756 $\pm$ 0.01	0.716 $\pm$ 0.00	—
GIN	$\mathcal{G}_1$	0.800 $\pm$ 0.01	0.704 $\pm$ 0.00	0.745 $\pm$ 0.01	0.717 $\pm$ 0.00	0.782 $\pm$ 0.00
	$\mathcal{G}_2$	0.748 $\pm$ 0.02	0.702 $\pm$ 0.00	0.743 $\pm$ 0.01	0.716 $\pm$ 0.00	—
GraphSAINT	$\mathcal{G}_1$	0.718 $\pm$ 0.01	0.797 $\pm$ 0.00	0.688 $\pm$ 0.01	0.750 $\pm$ 0.00	0.862 $\pm$ 0.01
	$\mathcal{G}_2$	0.702 $\pm$ 0.02	0.791 $\pm$ 0.01	0.694 $\pm$ 0.01	0.756 $\pm$ 0.00	—
SEAL	$\mathcal{G}_1$	0.818 $\pm$ 0.02	0.754 $\pm$ 0.01	0.834 $\pm$ 0.01	0.751 $\pm$ 0.00	0.887 $\pm$ 0.01
HONEM	$\mathcal{G}_1, \mathcal{G}_2$	0.697 $\pm$ 0.02	0.818 $\pm$ 0.00	0.589 $\pm$ 0.01	0.769 $\pm$ 0.00	0.815 $\pm$ 0.00
HO-GNN	$\mathcal{G}_2$	0.811 $\pm$ 0.00	0.789 $\pm$ 0.00	0.820 $\pm$ 0.02	0.879 $\pm$ 0.00	0.856 $\pm$ 0.00
DGE-concat	$\mathcal{G}_2$	<b>0.886</b> $\pm$ 0.01	0.815 $\pm$ 0.01	0.774 $\pm$ 0.01	0.802 $\pm$ 0.00	<b>0.910</b> $\pm$ 0.00
DGE-concat*	$\mathcal{G}_2$	0.856 $\pm$ 0.01	0.779 $\pm$ 0.01	0.712 $\pm$ 0.02	0.898 $\pm$ 0.00	0.904 $\pm$ 0.00
DGE-pool	$\mathcal{G}_2$	0.851 $\pm$ 0.02	<b>0.920</b> $\pm$ 0.00	0.838 $\pm$ 0.03	0.913 $\pm$ 0.00	0.898 $\pm$ 0.00
DGE-pool*	$\mathcal{G}_2$	0.845 $\pm$ 0.01	0.901 $\pm$ 0.00	0.802 $\pm$ 0.06	0.769 $\pm$ 0.01	0.815 $\pm$ 0.00
DGE-bag	$\mathcal{G}_2$	<b>0.887</b> $\pm$ 0.00	0.907 $\pm$ 0.00	<b>0.876</b> $\pm$ 0.01	<b>0.921</b> $\pm$ 0.00	0.895 $\pm$ 0.00
DGE-bag*	$\mathcal{G}_2$	0.862 $\pm$ 0.02	0.894 $\pm$ 0.00	0.856 $\pm$ 0.00	0.891 $\pm$ 0.00	0.891 $\pm$ 0.00
DGE-batch*	$\mathcal{G}_2$	0.853 $\pm$ 0.03	0.871 $\pm$ 0.01	0.830 $\pm$ 0.01	0.865 $\pm$ 0.00	0.865 $\pm$ 0.00

\* shared parameters

Table 9: Node classification results (micro F1) under various parameter budgets. Bold font indicates the best result for each budget and data set.

Model	Layers	Total parameters (Wiki)				Total parameters (Mag)			
		50k	100k	500k	1m	50k	100k	500k	1m
GCNII	2	0.60	0.60	0.60	0.60	0.78	0.79	0.79	0.79
	4	0.58	0.61	0.62	0.63	0.79	0.80	0.80	0.80
	8	0.60	0.61	0.62	0.62	0.77	0.79	0.79	0.79
GATv2	2	0.46	0.52	0.61	0.62	0.26	0.30	0.37	0.40
	4	0.40	0.47	0.51	0.55	—	—	—	—
GIN	2	0.59	0.61	0.62	0.62	0.82	0.82	0.83	0.83
	4	0.50	0.54	0.61	0.62	0.66	0.76	0.79	0.79
GraphSAINT	2	0.61	<b>0.64</b>	0.66	0.66	0.79	0.79	0.81	0.82
	4	0.55	0.62	0.66	0.66	0.54	0.65	0.70	0.74
DGE-bag	2	0.52	0.58	<b>0.68</b>	<b>0.69</b>	0.81	<b>0.84</b>	<b>0.86</b>	<b>0.87</b>
DGE-bag*	2	<b>0.63</b>	<b>0.64</b>	0.65	0.65	<b>0.83</b>	<b>0.84</b>	0.85	0.85

\* shared parameters

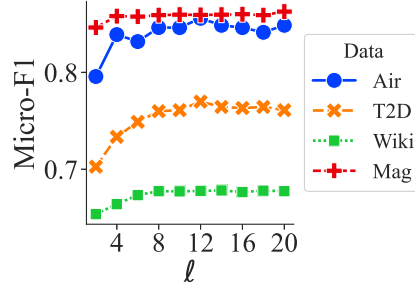


Figure 5: Node classification performance for DGE-bag as a function of the number of base learners ( $\ell$ ).

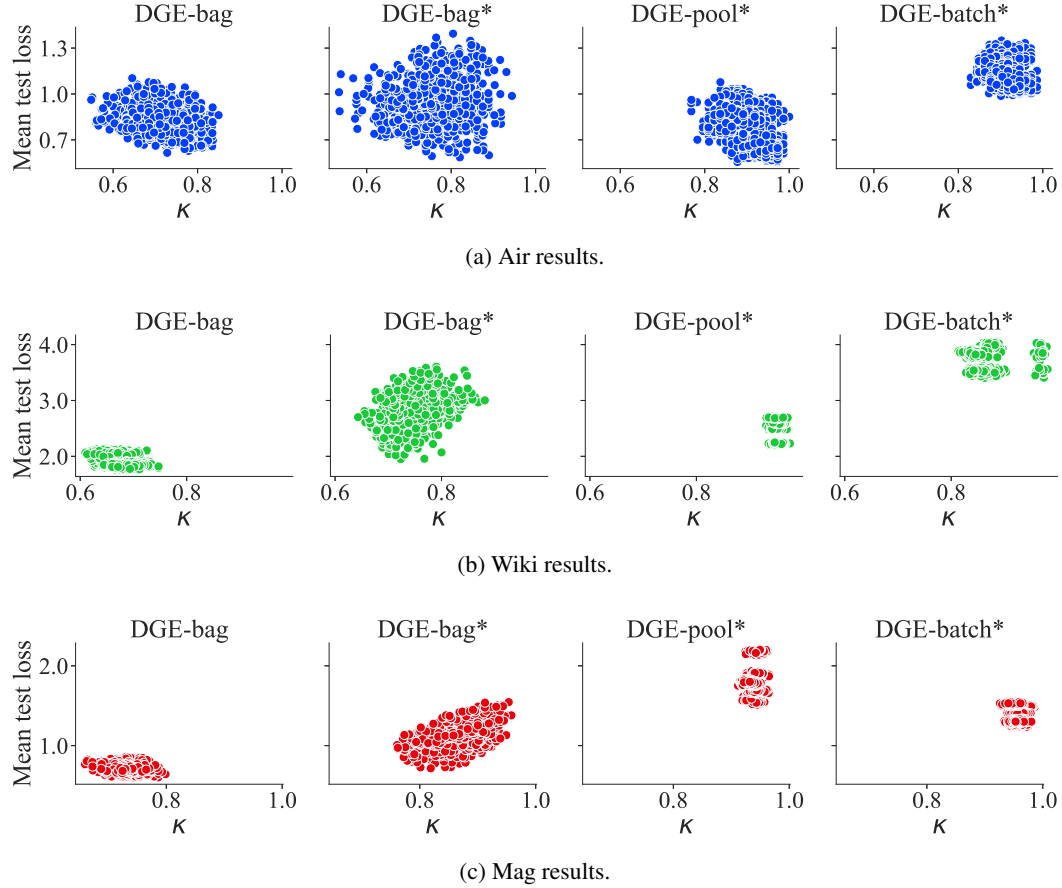


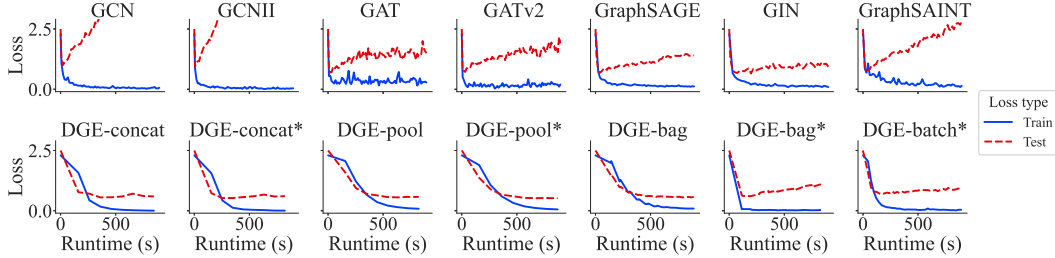
Figure 6: Mean node classification loss of each pair of classifiers, plotted as a function of Cohen's kappa (lower values indicate lower agreement). Each point represents one pairwise comparison between the 16 classifiers in each of the 5 testing folds. All plots contain the same number of points.

## F TRAINING TIME AND CONVERGENCE

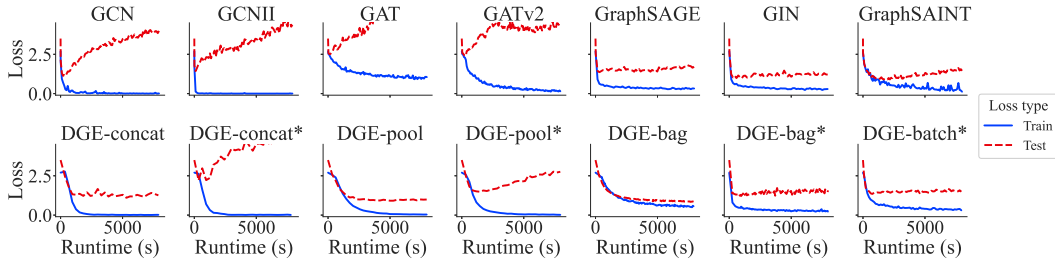
We analyzed the additional time cost incurred by DGE by plotting training and testing loss as a function of total runtime for Air and T2D. We used an NVIDIA GeForce GTX TITAN X GPU and dual 8-core 2.4GHz Intel Xeon processors for each model. As Figure 7 shows, DGE converged more slowly but generalized significantly better than all baselines. While the baselines typically converged quickly on the training set, they also rapidly began to overfit on the testing set. The increase in test error was sharper for the full-batch models like GCN and GCNII since they completed more epochs in a shorter period of time. DGE-bag, and, to a lesser extent, DGE-pool, generalized extremely well. These observations reflect overall model performance as reported in Table 2. One difference is that many of the baseline results reported in Table 2 used only a single GNN layer (Section 4.2). For the results in Figure 7, we fixed the number of GNN layers to 2 in order to confirm that the poor generalization of baselines (Section 4.2) was not due to lack of model capacity or underfitting the training set. This is why some models, like GCNII on Air, were relatively competitive in the main results but show high generalization error here.

The sample-based methods, including DGE, incurred significant overhead from neighbor sampling (over 95% of total runtime). This was exacerbated by the fact that all models preferred a high number of neighbor samples at the first layer, so repeatedly sampling a single neighbor at the second layer (Appendix C) produced substantial procedural overhead. DGE would train significantly faster with a more efficient mechanism for sampling these two-hop neighbors (such as pre-fetching).

All variants of DGE converged in fewer epochs than baselines, likely because DGE sees several subgraphs for each node during the same epoch (by virtue of being an ensemble) and because  $\mathcal{G}_k$  is sparser than  $\mathcal{G}_1$ . On T2D, convergence took approximately 100 epochs for DGE, 100-150 for minibatch baselines, and 1000-1500 for full-batch baselines. On other data sets, DGE generally converged in approximately 15-20 epochs, minibatch baselines converged in 20-30 epochs, and other full-batch baselines took 100-200 epochs (except Mag and Mag+, which took the full-batch models 500+ epochs).



(a) Air results.



(b) T2D results.

Figure 7: Node classification loss for 2-layer models on the first testing fold, shown as a function of training time. Runtime is measured as wallclock time from the start of the first epoch. High loss values are truncated to preserve scale. Similar trends held for all testing folds.

## G ANALYSIS OF NODE CHARACTERISTICS

To better understand DGE’s performance, we analyzed the node classification predictions by plotting the change in test loss for each node (Figure 8). We hypothesized that DGE would provide the strongest improvements on nodes with higher out-degree and larger higher-order families ( $|\Omega^2|$ ); however, this was not consistently true for all data sets. Instead, we found that a node’s homophily ( $\mathcal{H}$ , defined as the fraction of neighbors that have the same class) was the best indicator of performance improvement. This observation is noteworthy for future work, especially since most GNNs struggle to model graphs with low homophily (Zhu et al., 2020).

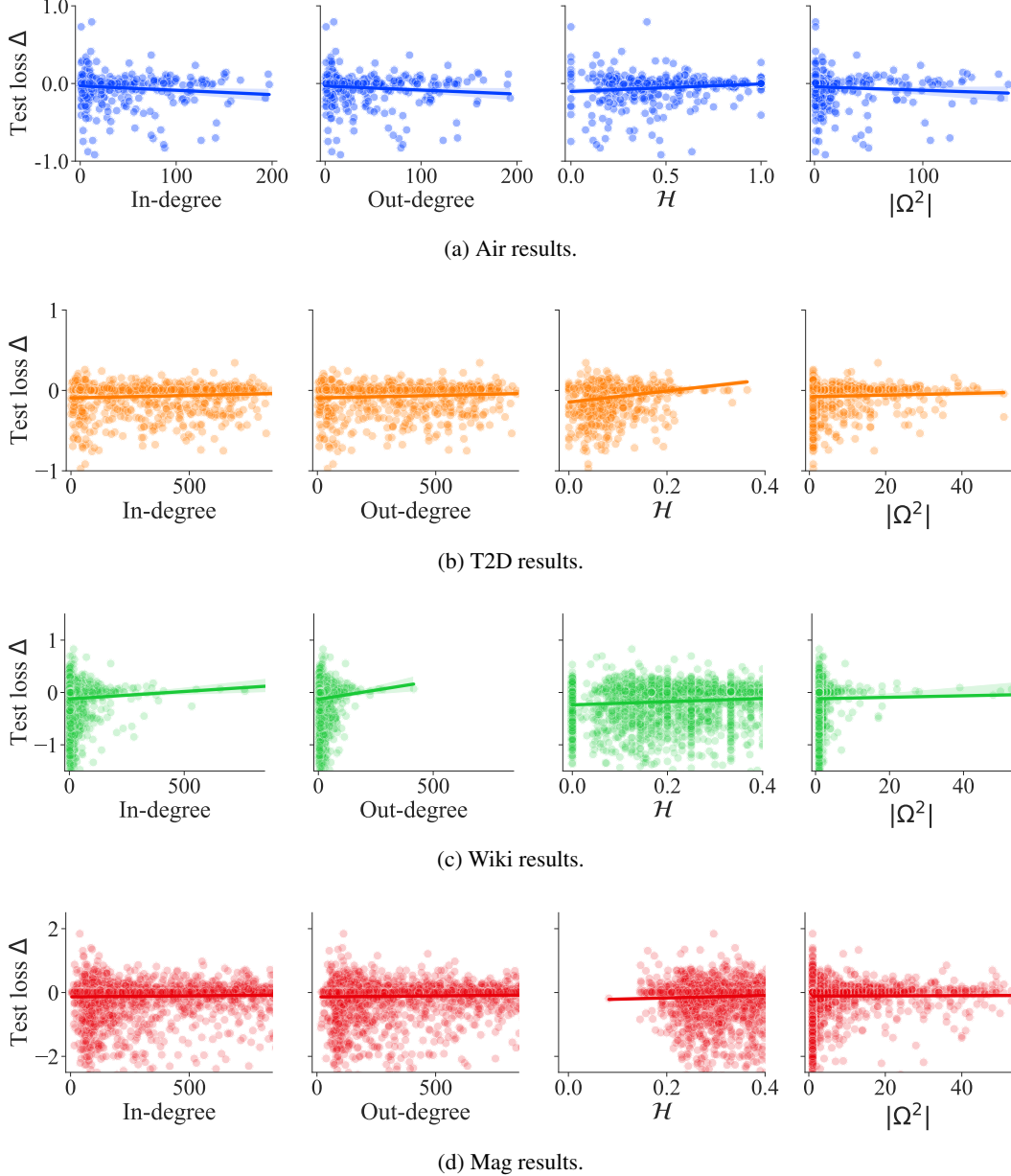


Figure 8: Delta in test loss for DGE-bag compared to GraphSAGE (the base GNN) for node classification (i.e., values  $< 0$  mean that GraphSAGE produced higher loss than DGE-bag), plotted as a function of key node features in  $\mathcal{G}_1$ . Out-degree and in-degree are unweighted. Each point represents a node in the test set. All five testing folds are included. Shading represents a 95% CI for the regression line.

## H ADDITIONAL CONSIDERATIONS ON HON CONSTRUCTION

We used GrowHON (Krieg et al., 2020a) to construct all HONs. GrowHON provides a parameter  $\tau$ , which helps regularize the graph by raising the KL-divergence threshold required for a new higher-order node to be created, and we found that the choice of  $\tau$  impacted DGE’s success in each task. Figure 9 shows the results of varying  $\tau$  for each data set and task. In general, the optimal  $\tau$  value varied between data sets for classification tasks, while for link prediction, a lower  $\tau$  generally yielded better results. We did not tune DGE’s hyperparameters for these experiments, so it is likely that further tuning would reduce the difference between each model. However, these observations have important consequences for future work, especially with respect to integrating prior studies on graph structure learning, which have argued that different graph representations may be useful for different predictive tasks (Brugere & Berger-Wolf, 2020). Ideally, we could incorporate HON construction into the learning framework so that the model could learn the structure that is best-suited to each task.

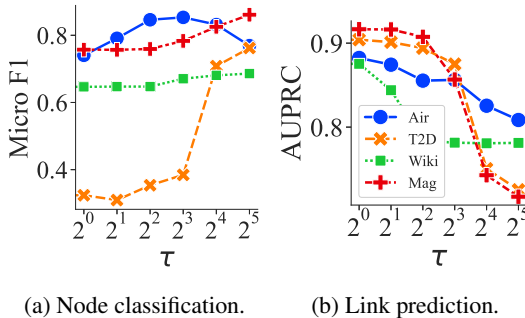


Figure 9: Performance as a function of  $\tau$ . Each point represents the mean of all 5 testing folds.