DEEP ENSEMBLES FOR GRAPHS WITH CONCISE HIGHER-ORDER REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) can be limited by the expressiveness of first-order graph representations when applied to data with higher-order path dependencies. Although conventional higher-order networks (HONs) preserve path context, they introduce many predecessor-specific state nodes, thereby increasing model complexity. To address this issue, we propose CDGE, which combines a regularized low-rank concise higher-order network (CHON) representation with an ensemble of independently trained GNN learners. This design compresses the state space while preserving differences in state-conditioned neighborhoods. We evaluate CDGE on link prediction and node classification tasks using three real-world datasets: Air, Ship, and Journal. Under comparable parameter budgets, CDGE generally outperforms existing GNN baselines on both tasks while substantially reducing the numbers of nodes and edges in the HON representation. Further analyses show that predictive diversity and accuracy among independent learners, together with a moderate ensemble size, are key to CDGE's effectiveness. These findings provide useful guidance for the design of GNN ensembles for data with higher-order path dependencies. Code is available at `https://anonymous.4open.science/r/sr-cdge-reproducible-code-071F`.

1 INTRODUCTION

Graph neural networks (GNNs) propagate and aggregate information over local graph neighborhoods and have become standard tools for graph learning tasks such as node classification and link prediction (Kipf & Welling, 2017; Hamilton et al., 2017). Their use generally assumes that the observed graph provides an adequate approximation of the underlying neighborhood structure. This assumption is questionable for systems observed as paths or trajectories, where the next transition may depend not only on the current node but also on the node or nodes visited immediately before it. Such higher-order dependencies arise in transportation, mobility, citation, and other networked systems (Rosvall et al., 2014; Xu et al., 2016; Saebi et al., 2020). When these interactions are collapsed into a first-order graph, the resulting representation can introduce transitive paths that were not observed and merge local neighborhoods with different historical contexts. A first-order graph may therefore fail to describe the state-conditioned neighborhoods that govern the data.

Higher-order networks (HONs) address this limitation by introducing conditional state nodes that preserve path context (Xu et al., 2016; Rosvall et al., 2014; Saebi et al., 2020). For example, a second-order state $j \mid i$ denotes being at node j after arriving from node i. Unlike a first-order network, which records only pairwise adjacency, an HON can represent transitions conditioned on predecessor nodes and has been used to model human mobility, flow systems, disease progression, and citation networks. However, conventional HON construction typically materializes separate predecessor-specific state nodes for distinct conditional contexts. As the number of physical nodes and the path order increase, a single physical node may correspond to many state nodes, substantially enlarging the graph and increasing the cost of graph construction, neighborhood sampling, and downstream GNN training. Moreover, integrating the representations of multiple state nodes back into a stable physical-node representation remains challenging.

Krieg et al. showed that the state nodes associated with one physical node in an HON can be interpreted as distinct neighborhood subspaces. Their Deep Graph Ensemble (DGE) models these state-conditioned neighborhoods with independently trained GNN learners, allowing the ensemble

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to exploit higher-order neighborhood variation (Krieg et al., 2023). Related path-aware GNNs also use paths or random walks to improve spatial representations (Eliasof et al., 2022; Jin et al., 2022). Nevertheless, DGE still relies on the full predecessor-specific HON representation. When the HON contains many conditional states, training multiple GNN learners over the complete state space incurs substantial storage, sampling, and computational costs. More fundamentally, it remains unclear how to remove redundant HON states while preserving the neighborhood differences that are useful for downstream prediction.

Recent concise network models provide a complementary route to this problem. By regularizing transition statistics and applying convex non-negative matrix factorization, such models represent predecessor-conditioned behavior with a small number of interpretable states (Sahasrabuddhe et al., 2025; Ding et al., 2010). This construction can reduce the state space while retaining recurrent transition patterns, but its use as the computational substrate of a deep graph ensemble has not been established. The central challenge is therefore to compress the higher-order representation without eliminating the state-conditioned variation on which ensemble learning depends.

Here we introduce the Concise Deep Graph Ensemble (CDGE), which combines a regularized low-rank concise higher-order network (CHON) representation with an ensemble of independently trained GNN learners. CDGE first regularizes predecessor-conditioned transition distributions and uses convex non-negative matrix factorization to group similar transition behaviors into a small set of behavioral states. It then samples states from these concise state families and uses independent GNN learners to model their state-conditioned neighborhoods. State-family aggregation and residual gating combine the learner representations, and probability averaging produces the final prediction. This design is intended to reduce the state-space and graph-size overhead of HONs while retaining complementary higher-order neighborhoods.

We ask whether a low-rank CHON representation can substantially reduce the numbers of HON nodes and edges while preserving the state-conditioned variation required by a deep GNN ensemble. We study this question on three real-world path-based datasets: Air, Ship, and Journal. Link prediction is the primary task, whereas node classification serves as an auxiliary test of whether the shared state representation retains predictive structural information. CDGE achieves strong link-prediction performance on Air and Ship while substantially compressing the HON representation; the Journal results reveal a stability boundary for small and sparse path data. Further analyses show that predictive diversity and accuracy among independent learners, together with a moderate ensemble size, are central to CDGE performance.

Our main contributions are as follows:

- We introduce CDGE, a deep graph ensemble for data with higher-order path dependencies. CDGE replaces a large collection of predecessor-specific HON states with regularized low-rank CHON states and models their state-conditioned neighborhoods with independently trained GNN learners.

- We develop an end-to-end pipeline from path data and transition statistics to CHON state compression and downstream GNN ensemble prediction. A flow-overlap criterion selects the number of concise states independently for each physical node.

- We evaluate CDGE on Air, Ship, and Journal. The link-prediction experiments show strong performance on Air and Ship together with substantial reductions in HON node and edge counts, while the node-classification experiments provide auxiliary evidence for the usefulness of the compressed state representation.

- We analyze CDGE with respect to ensemble size, prediction diversity, network depth, node characteristics, convergence, and the CHON overlap threshold. These analyses identify when moderate ensemble sizes balance predictive performance and computational cost, and expose the stability limitations that arise on small and sparse datasets.

2 RELATED WORK

2.1 HIGHER-ORDER NETWORKS

Higher-order networks represent historical dependencies in observed paths. Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a set of paths, where $S_i = \langle s_{i1}, \ldots, s_{im_i} \rangle$ records the order in which entities occur, and let $\mathcal{A}$ be the set of entities appearing in all paths. If the probability of moving to k after arriving at node j from predecessor i differs from the probability estimated from j alone, that is, $P(k \mid j, i) \neq P(k \mid j)$, then the current node is insufficient to describe the next transition; the history by which it was reached must also be retained. The graph representing $\mathcal{S}$ is not unique. We consider a first-order network (FON), the higher-order network (HON) introduced by Xu et al. (2016), and the concise higher-order network (CHON) proposed by Sahasrabuddhe et al. (2025).

A first-order network $G_1 = (V_1, E_1, w_1)$ has $V_1 = \mathcal{A}$. If u and v occur consecutively in at least one path, then $(u, v) \in E_1$; the edge weight $w_1(u, v)$ records the observed frequency of that transition. Consequently, the first-order transition probability from j to k, $P_1(k \mid j) = w_1(j, k) / \sum_{k'} w_1(j, k')$, is independent of the predecessor from which j was reached. A FON retains direct transitions but merges subsequent behaviors under different histories into a single distribution.

A higher-order network $G_k = (V_k, E_k, w_k)$ distinguishes path histories through conditional nodes. A conditional node can be written as $u' = \langle a_1, \ldots, a_m \rangle$, where $m \leq k$ and the final entity a_m is its corresponding physical node; the mapping $f(u') = a_m$ specifies this correspondence. For example, in the second-order case, $j \mid i$ denotes the state of having arrived at j from i. All nodes corresponding to j form its higher-order node family $\Omega_j^k = \{u' \in V_k : f(u') = j\}$. Conditional nodes within this family can have different weighted neighborhoods, encoding transitions under different histories. The on-demand construction of Xu et al. (2016) does not create a conditional node for every possible predecessor; it retains histories that provide additional information about transitions.

The conditional labels of an HON identify the specific predecessors associated with its states, but do not directly summarize transition patterns shared by similar predecessors. For example, $j \mid i_1$ and $j \mid i_2$ may have similar subsequent behaviors, yet their separate conditional nodes do not express this commonality. Retaining more predecessor-specific states also increases the numbers of nodes and edges; when some predecessors are observed only a few times, their conditional transition estimates may be unreliable. Thus, an HON identifies path histories but does not readily explain the overall transition behavior of a physical node using a small number of states.

A CHON uses behavioral pattern states to summarize similar predecessor-conditioned behaviors. For each physical node j, Sahasrabuddhe et al. (2025) first regularize the transition counts of path triples $i \to j \to k$ to obtain a target matrix $X^{(j)}$; they then approximate it with r_j pattern states, such that $\widehat{X}_{ki}^{(j)} = \sum_{\alpha=1}^{r_j} \widehat{P}(\alpha \mid j, i) \widehat{P}(k \mid j, \alpha)$. Here, $\widehat{P}(\alpha \mid j, i)$ is the probability of entering pattern α after arriving at j from i, and $\widehat{P}(k \mid j, \alpha)$ is the probability of moving from that pattern to k. Convex nonnegative matrix factorization constrains the outgoing transition distributions of the patterns to be convex combinations of observed behaviors, while flow overlap determines the number of states r_j for each node. A pattern state can therefore summarize several predecessors with similar transition behaviors, rather than correspond to only one specific predecessor.

Determining which conditional behaviors to retain, how many state nodes to create, and how to connect them is therefore a nontrivial problem in constructing a graph from paths. For algorithmic details, see Sahasrabuddhe et al. (2025); the CHON construction procedure used here follows their work.

Related graph-based models. The term "higher-order" can refer to distinct structural and learning problems. Hypergraphs capture multi-way associations within samples (Lin et al., 2026); higher-order topology can guide graph generation (Huang & Birdal, 2026); and motifs serve as structural units for community detection (Xiao et al., 2025). Higher-order representation learning for links and other multi-node targets instead extracts structural information between candidate nodes from a given graph (Xu & Zou, 2024). These problems differ from the historical dependencies in observed paths studied here.

HONs address path representation before graph learning. They construct conditional nodes from ordered observed paths so that different histories leading to the same physical node can induce different subsequent transitions (Xu et al., 2016; Saebi et al., 2020; Krieg et al., 2021; Fu et al., 2024; Li et al., 2025; Zhang et al., 2026). Sahasrabuddhe et al. (2025) instead compress second-order path representations by using a small number of states to summarize similar predecessor-conditioned behaviors, rather than assigning each retained predecessor its own state.

Structural encodings on a given graph complement graph construction from observed paths. Eliasof et al. (2023) generate positional encodings using graph-dependent propagation operators, while Canturk et al. (2024) learn transferable positional and structural encodings. Path-based aggregators using random walks can also improve GNN expressiveness (Eliasof et al., 2022; Jin et al., 2022), but their sampled paths are still determined by the input graph. If aggregation into a first-order graph has discarded predecessor-conditioned information from observed paths, enriching structural encodings on that graph cannot recover the original path-transition distribution.

Deep Graph Ensemble (DGE) applies conditional HON neighborhoods to GNN ensemble learning (Krieg et al., 2023). However, DGE exploits neighborhood diversity without grouping predecessors with similar behaviors into a small number of interpretable transition modes. Retaining many conditional states can increase the size of the graph representation and its sampling burden; the original study also reports substantial runtime overhead from neighbor sampling. Building on CHON, we propose CDGE to study downstream performance while examining whether redundant predecessor-specific states can be reduced without discarding essential transition differences.

2.2 Related graph-based models

Graph neural networks (GNNs) aggregate neighborhood information on a given graph and update node representations layer by layer. Let z denote a node in the graph. A generic message-passing procedure can be written as

$$\mathbf{h}_z^{(t)} = \mathrm{COMBINE}_t\Big(\mathbf{h}_z^{(t-1)}, \mathrm{AGGREGATE}_t\Big(\{\mathbf{h}_w^{(t-1)} : w \in \mathcal{N}(z)\}\Big)\Big). \tag{1}$$

Here, $\mathcal{N}(z)$ denotes the neighborhood of node z. The graph structure determines which neighborhoods are available for message passing, whereas the encoder determines how information from those neighborhoods is aggregated. Recent positional and structural encoding methods further enrich node representations on a given graph (Eliasof et al., 2023; Canturk et al., 2024). The distinction relevant to this work is that a single physical node may correspond to multiple state nodes with different conditional neighborhoods. Thus, the GNN learns representations of state nodes, whereas node classification ultimately requires a prediction for each physical node.

Deep Graph Ensembles (DGE) exploit differences among the conditional neighborhoods associated with the same physical node in a higher-order network. DGE samples states from the higher-order state family of each node, trains multiple base GNNs, and combines their outputs (Krieg et al., 2023). This design allows the ensemble to use history-dependent variation that cannot be represented by a single first-order neighborhood. However, DGE primarily exploits neighborhood diversity on an already constructed HON; it does not aim to group predecessor states with similar successor behavior into a small number of transition modes. When many conditional states are retained, the state graph and the associated sampling operations may become more costly. The original DGE study also reported substantial neighbor-sampling overhead in its implementation. The issue, therefore, is not insufficient GNN depth, but whether the state representation used for ensembling contains redundant structure.

We propose Concise Deep Graph Ensemble (CDGE), a deep ensemble method based on a concise higher-order graph. Learner ℓ samples a state $z_u^{(\ell)}$ for physical node u from $\mathcal{F}(u)$ and computes its representation from the corresponding conditional neighborhood in the concise graph. The state choices made by different learners define distinct training views, but they do not imply the construction of L graphs with different edge sets. The base encoders may use either independent or shared parameters. Their state representations or class probabilities are combined using bag, pool, or concat aggregation, and are then mapped back to predictions for the physical-node labels. Our current implementation uses a directed GraphSAGE encoder that processes self, incoming-neighbor, and outgoing-neighbor information separately, but the CDGE state-sampling and ensembling procedures do not depend on this particular encoder.

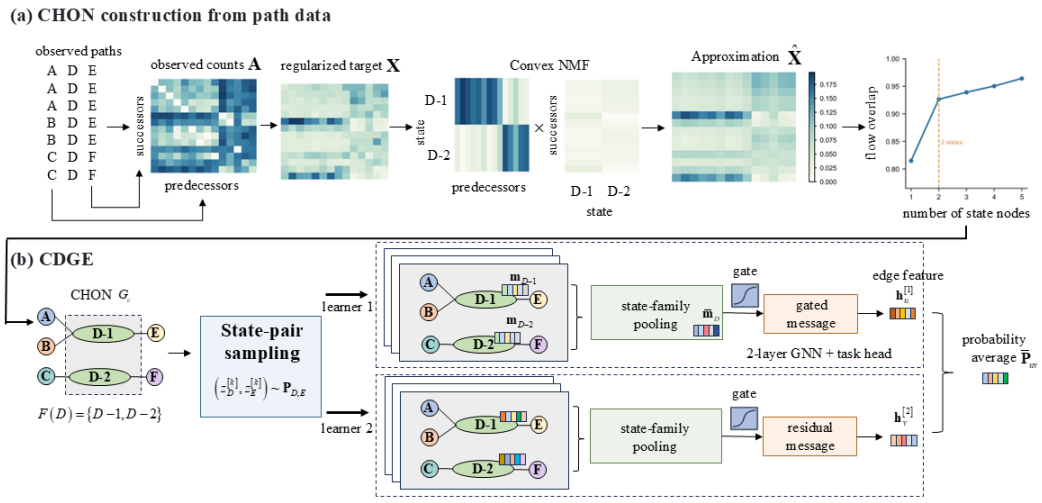

Figure 1: Overview of CDGE. Observed paths are converted into a regularized low-rank CHON representation. Independent GNN learners sample state-conditioned neighborhoods, apply state-family residual gating, and average their output probabilities for the final prediction.

3 METHOD

We refer to the proposed model as the *Concise Deep Graph Ensemble* (CDGE). CDGE separates higher-order graph construction from downstream representation learning. Given observed paths, it first builds a regularized low-rank concise higher-order network (CHON), then trains independent GNN learners on state-conditioned neighborhood views from the same graph. Each learner produces task-specific predictions from a sampled state representation, and the final CDGE-bag prediction is obtained by averaging learner probabilities. The full construction equations, sampling rules, losses, and implementation settings are provided in Appendix 7. An overview of the complete pipeline is shown in Figure 1.

3.1 PROBLEM SETTING AND CHON INPUT

Let $\mathcal{S}$ be a collection of observed paths over physical nodes V. A higher-order representation contains a state set Z, where several states may correspond to the same physical node. The map $b : Z \to V$ identifies the physical node of each state, and its state family is $\mathcal{F}(D) = \{z \in Z : b(z) = D\}$ for physical node $D \in V$. States in one family inherit the feature and supervision of D, but can have different incoming and outgoing neighborhoods. CDGE therefore attaches labels to physical nodes while performing message passing on sampled state nodes.

For each physical node j, the path data provide predecessor-successor counts $A^{(j)}_{ki}$ for triples $i \to j \to k$. We regularize the corresponding predecessor-conditioned transition columns toward a memoryless distribution using a Dirichlet prior. In compact form, the regularized target is $X^{(j)}_{ki} = (A^{(j)}_{ki} + \mu_j M^{(1,j)}_k)/(n^{(j)}_i + \mu_j)$, where $n^{(j)}_i = \sum_k A^{(j)}_{ki}$ and μ_j is selected by leave-one-out likelihood. The resulting weighted directed graph is denoted by $G_c = (Z, E_c, w_c)$.

3.2 CONCISE STATE CONSTRUCTION

Rather than materializing one state for every retained predecessor, CDGE approximates each regularized matrix with a small number of behavioral states using convex non-negative matrix factorization: $X^{(j)} \approx \widehat{X}^{(j)}_{\text{out}} (\widehat{X}^{(j)}_{\text{in}})^\top$. The outgoing factor describes the successor distribution of a state, whereas the incoming factor assigns a predecessor context to that state. Convexity keeps the learned transition modes within the support of the regularized observations.

The number of states is selected independently for each physical node. CDGE chooses the smallest rank whose count-weighted flow-overlap score reaches a target θ, subject to a maximum rank. Rank

one gives a memoryless representation, while larger ranks retain additional predecessor-conditioned variation. Local factors are then stitched into a global CHON: an outgoing factor from one physical node is combined with the incoming factor of the next physical node to produce a state-level edge weight. Edges below a relative-weight threshold σ are removed. This construction reduces redundant state nodes while retaining transition modes that are supported by the observed paths.

3.3 State-conditioned GNN encoder

For each physical node D, learner k samples one state $z_D^{[k]}$ from $\mathcal{F}(D)$, with probabilities proportional to state out-degree; the corresponding in-degree distribution is used for incoming views. Each sampled state is expanded into separate incoming and outgoing one- and two-hop neighborhoods. The encoder keeps these directions separate and samples neighbors according to edge weights. Every state inherits the identity feature of its physical node, so the state representation differs through its sampled, path-conditioned neighborhoods.

At each layer, incoming and outgoing neighbor messages are computed by directed mean aggregation and concatenated. For state z with physical node $D = b(z)$, CDGE computes a family mean over the states in $\mathcal{F}(D)$ and a state-specific residual. The gated message is expressed inline as $\widetilde{\mathbf{m}}_z = \overline{\mathbf{m}}_D + g_z(\mathbf{m}_z - \overline{\mathbf{m}}_D)$. Thus, the gate interpolates between a family-shared structural component and a state-specific neighborhood signal. A learned scalar gate is initialized at the neutral value $g_z = 0.5$ and is learned independently by each learner. The gated message is combined with the state self-feature before the GNN update.

After the final layer, the sampled root state gives a physical-node embedding $\mathbf{h}_D^{[k]}$. This shared encoder is task-agnostic: a node-classification head maps the embedding to class probabilities, whereas a link-prediction head forms the endpoint representation $[\mathbf{h}_u \odot \mathbf{h}_v; |\mathbf{h}_u - \mathbf{h}_v|]$ before producing an edge logit.

3.4 Independent learners and prediction

CDGE-bag uses K learners with independent parameter initialization, state samples, neighborhood-sampling streams, and gate parameters. All learners share the same CHON and physical-node supervision, but they do not share their learned weights. At inference, learner probabilities are averaged before the task metric is computed. This preserves the interpretation of each learner as an independent state-conditioned view and provides prediction diversity without a learned ensemble router.

Node classification is optimized with the physical-node labels and a softmax head. Link prediction uses family-level masking, state-pair sampling, a binary cross-entropy objective, and continuous sigmoid probabilities for AUPRC. These details, together with the parameter-sharing variants and default implementation settings, are given in Appendix 7 and the evaluation protocol in Section 7.16.

4 Experiments

4.1 Data and evaluation protocol

We evaluate CDGE on three path-based datasets: Air, Ship, and Journal. Air contains domestic air-travel trajectories, Ship contains ordered shipping trajectories, and Journal represents ordered citation sequences between journals. Nodes are physical entities, while consecutive records define directed transitions and predecessor-conditioned states. The symbols V, E, and H in Table 1 denote the numbers of nodes, edges, and average edge homophily, respectively. Link prediction is the primary task and is evaluated with AUPRC; node classification is an auxiliary task evaluated with micro-F1. The full split, masking, sampling, and implementation protocols are provided in Appendix 7.

4.2 Graph construction and compression

The released benchmarks provide higher-order edge lists. We recover the available second-order transition counts, construct regularized CHON states with leave-one-out prior selection and flow-overlap rank selection, and trim low-weight edges. Table 1 summarizes the resulting graph sizes.

Table 1: Dataset classes and graph sizes for the first-order (FON), higher-order (HON), and concise higher-order (CHON) representations. V, E, and H denote the number of nodes, the number of edges, and average edge homophily, respectively.

Dataset	Classes	FON			HON			CHON		
		V	E	H	V	E	H	V	E	H
Air	10	435	6,422	0.55	3,632	32,341	0.47	1,449	11,995	0.34
Ship	36	621	4,840	0.58	4,637	34,850	0.58	1,455	16,032	0.36
Journal	3	22	159	0.26	158	1,289	0.13	27	232	0.22

Table 2: Link-prediction and node-classification results on Air, Ship, and Journal. AUPRC denotes the area under the precision–recall curve for link prediction, whereas micro-F1 denotes the micro-F1 score for node classification. Values are reported as mean ± standard deviation. Bold and underlined entries denote the best and second-best results within each task and dataset, respectively. An asterisk marks the shared-parameter variant.

Method	Air		Ship		Journal	
	AUPRC	micro-F1	AUPRC	micro-F1	AUPRC	micro-F1
GCN	0.826 ± 0.02	0.724 ± 0.02	0.700 ± 0.02	0.705 ± 0.02	0.769 ± 0.02	0.0500 ± 0.02
GCNII	0.866 ± 0.01	0.683 ± 0.01	0.700 ± 0.01	0.725 ± 0.01	0.783 ± 0.01	0.542 ± 0.03
GAT	0.756 ± 0.02	0.731 ± 0.02	0.810 ± 0.02	0.700 ± 0.02	0.585 ± 0.02	0.363 ± 0.05
GATv2	0.790 ± 0.02	0.754 ± 0.01	0.804 ± 0.02	0.744 ± 0.01	0.628 ± 0.02	0.458 ± 0.04
GraphSAGE	0.809 ± 0.01	0.724 ± 0.03	0.701 ± 0.02	0.643 ± 0.02	0.632 ± 0.01	0.548 ± 0.05
GIN	$\underline{0.853 \pm 0.02}$	0.710 ± 0.02	0.714 ± 0.01	0.618 ± 0.02	$\underline{0.844 \pm 0.02}$	0.405 ± 0.04
GraphSAINT	0.831 ± 0.03	0.759 ± 0.01	0.789 ± 0.03	0.754 ± 0.01	0.781 ± 0.03	0.458 ± 0.03
PathGCN	0.657 ± 0.02	$\underline{0.795 \pm 0.02}$	0.748 ± 0.02	0.618 ± 0.03	0.352 ± 0.02	0.411 ± 0.02
HONEM	0.664 ± 0.01	0.738 ± 0.01	0.868 ± 0.01	0.681 ± 0.02	0.393 ± 0.02	0.500 ± 0.03
HO-GNN	0.852 ± 0.02	0.772 ± 0.02	0.863 ± 0.02	0.643 ± 0.01	0.798 ± 0.01	0.411 ± 0.04
DGE-bag	0.856 ± 0.01	0.772 ± 0.01	0.812 ± 0.02	0.763 ± 0.02	0.827 ± 0.02	0.411 ± 0.05
DGE-bag*	0.853 ± 0.02	0.772 ± 0.02	0.885 ± 0.01	0.763 ± 0.01	0.755 ± 0.01	0.452 ± 0.05
DGE-pool	0.798 ± 0.02	0.759 ± 0.02	0.820 ± 0.02	$\underline{0.792 \pm 0.01}$	0.804 ± 0.02	0.417 ± 0.04
DGE-pool*	0.793 ± 0.01	0.745 ± 0.01	0.870 ± 0.01	0.773 ± 0.02	0.738 ± 0.01	0.458 ± 0.03
DGE-concat	0.808 ± 0.02	0.703 ± 0.03	0.828 ± 0.02	0.754 ± 0.02	0.766 ± 0.02	0.506 ± 0.03
DGE-concat*	0.799 ± 0.03	0.697 ± 0.02	0.842 ± 0.02	$\underline{0.792 \pm 0.03}$	0.776 ± 0.03	0.458 ± 0.04
DGE-batch*	0.827 ± 0.01	0.738 ± 0.01	0.873 ± 0.01	0.749 ± 0.02	0.702 ± 0.01	0.548 ± 0.05
CDGE-bag	$\underline{0.924 \pm 0.01}$	$\mathbf{0.848 \pm 0.02}$	0.900 ± 0.02	$\mathbf{0.795 \pm 0.02}$	0.805 ± 0.02	$\mathbf{0.643 \pm 0.04}$
CDGE-bag*	0.922 ± 0.02	$\underline{0.828 \pm 0.01}$	0.890 ± 0.01	0.758 ± 0.01	0.713 ± 0.01	$\underline{0.589 \pm 0.04}$
CDGE-pool	$\mathbf{0.925 \pm 0.02}$	0.800 ± 0.01	0.890 ± 0.02	0.758 ± 0.01	0.795 ± 0.02	$\mathbf{0.643 \pm 0.03}$
CDGE-pool*	0.914 ± 0.01	0.779 ± 0.02	0.895 ± 0.01	$\underline{0.792 \pm 0.02}$	0.794 ± 0.01	$\underline{0.589 \pm 0.03}$
CDGE-concat	0.920 ± 0.01	0.786 ± 0.02	$\underline{0.905 \pm 0.01}$	0.739 ± 0.02	$\mathbf{0.897 \pm 0.02}$	$\mathbf{0.643 \pm 0.05}$
CDGE-concat*	0.910 ± 0.02	0.669 ± 0.04	$\mathbf{0.912 \pm 0.02}$	0.744 ± 0.03	0.815 ± 0.01	$\underline{0.589 \pm 0.05}$
CDGE-batch*	0.913 ± 0.03	0.800 ± 0.01	0.904 ± 0.03	0.749 ± 0.02	0.809 ± 0.03	$\underline{0.589 \pm 0.04}$

CHON reduces HON state nodes by approximately 60.1% on Air, 68.6% on Ship, and 82.9% on Journal, while reducing edges by 62.9%, 54.0%, and 82.0%, respectively. The amount of compression is therefore dataset-dependent, but all three configurations reduce both state and edge counts.

4.3 Downstream results

CDGE obtains the best AUPRC on Air and Ship, reaching 0.925 and 0.912, respectively. On Journal, the best CDGE configuration reaches 0.897, above the strongest first-order baseline at 0.844. For node classification, CDGE attains the best or tied-best micro-F1 on all three datasets, although the best variant depends on the dataset and task.

4.4 Parameter budgets and diagnostic figures

The parameter-budget experiment tests whether CDGE remains competitive at comparable model sizes. The horizontal layout in Table 3 reports the complete budget comparison; the detailed interpretation of budget dependence and learner-count sensitivity is given in Appendix 11.

The budget comparison is evaluated with a fixed two-layer backbone and a constrained parameter allocation, whereas Table 2 reports the main configurations. The two tables therefore need not have identical absolute values. The budget results show that CDGE-bag improves as the available capacity increases on Air and Ship, while Journal remains more variable because of its smaller and sparser candidate-link set.

Table 3: Link-prediction results under approximately matched parameter budgets. Each entry reports the mean AUPRC for a two-layer model. Air and Ship use five-fold cross-validation, whereas Journal uses its provided train/test candidate-link split. Values are rounded to two decimal places, and the best result within each budget and dataset is shown in bold.

	Air					**Ship**					**Journal**				
Method	10k	50k	100k	500k	1m	10k	50k	100k	500k	1m	10k	50k	100k	500k	1m
GCNII	0.68	0.81	**0.86**	0.85	0.85	0.70	0.73	0.75	0.73	0.74	0.40	**0.52**	**0.60**	0.70	0.78
GATv2	0.50	0.50	0.52	0.61	0.72	0.53	0.56	0.68	0.74	0.76	**0.50**	0.49	0.55	0.60	0.63
GIN	0.82	0.82	0.76	0.76	0.72	0.69	0.67	0.65	0.68	0.71	0.44	0.49	**0.60**	0.75	**0.84**
GraphSAINT	0.45	0.71	0.64	0.45	0.64	0.50	0.57	0.62	0.72	0.78	**0.50**	0.47	0.55	0.70	0.78
CDGE-bag	**0.84**	**0.84**	0.85	**0.87**	**0.88**	**0.79**	**0.79**	**0.80**	**0.82**	**0.83**	0.47	0.47	0.55	**0.76**	0.80

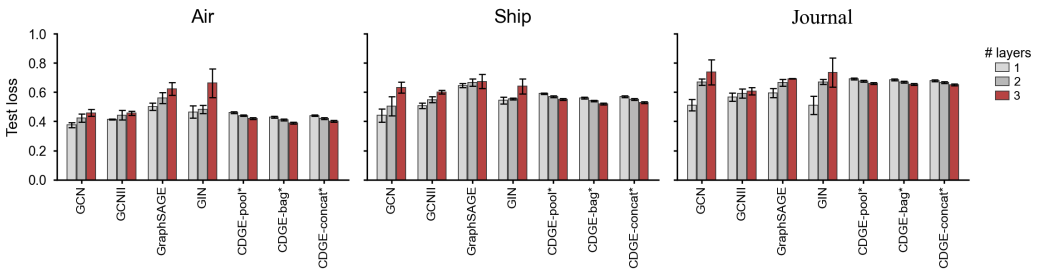

Figure 2: Depth sensitivity of link prediction on Air, Ship, and Journal. Bars show the test binary cross-entropy loss for one-, two-, and three-layer models; error bars denote standard deviations across three random seeds. Lower values indicate better link-prediction performance.

Figure 2 shows that ordinary GNN baselines generally become less stable as depth increases, whereas CDGE variants maintain or reduce test loss across the tested depths. The detailed loss interpretation and convergence discussion are provided in Appendix 11.

Figure 3 indicates that independently parameterized learners provide complementary predictions while retaining competitive pairwise loss. The detailed diversity analysis and dataset-specific boundary cases are given in Appendix 11.

5 Conclusion

We introduced CDGE, which trains a deep graph ensemble on regularized, interpretable low-rank second-order states. Across Air, Ship, and Journal, the reported CHON configurations reduce HON state nodes by approximately 60–83% and HON edges by approximately 54–82%. Link prediction is the primary evaluation: CDGE obtains the strongest results on Air and Ship and remains competitive on Journal, while auxiliary node classification confirms that the compressed representations retain useful state information.

The additional analyses show why the gains are not uniform. Moderate ensemble sizes usually capture most of the available benefit, independent learners provide useful prediction diversity, and the response to the CHON threshold depends on both the dataset and the task. Journal is the clearest boundary case because its smaller and sparser evaluation setting produces larger variation across learners and thresholds. Overall, CDGE provides a compact and interpretable substrate for higher-order graph ensembles, but its final configuration should be selected with the downstream task and dataset explicitly in view.

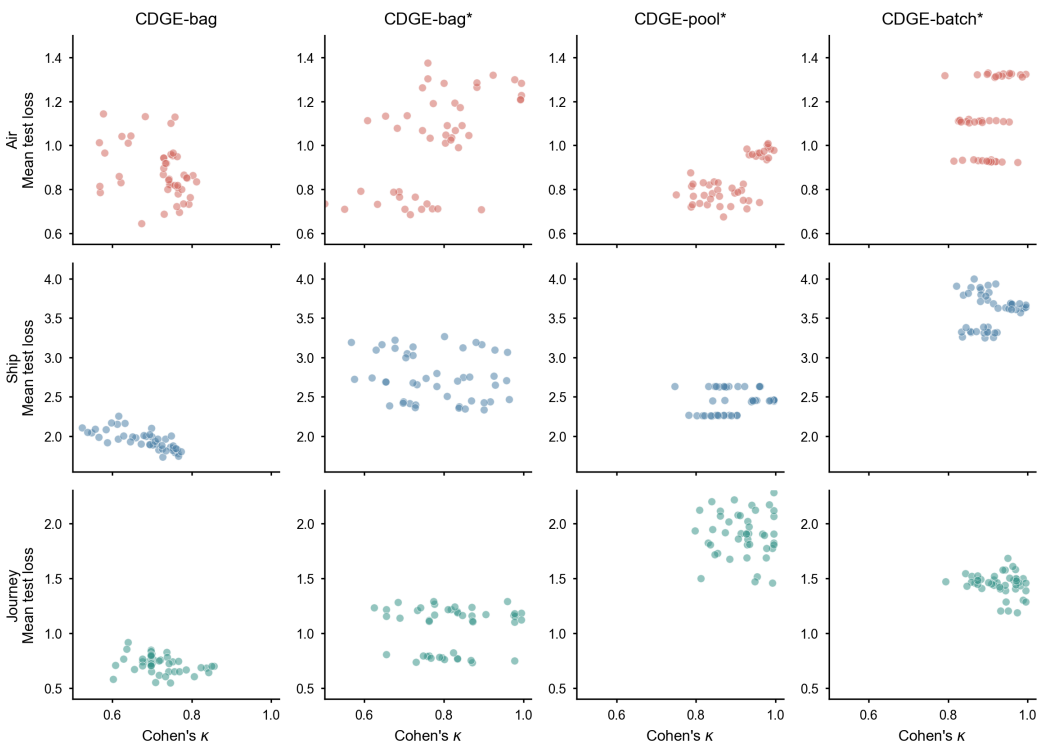

Figure 3: Accuracy–diversity relationship among CDGE learners on Air, Ship, and Journal. Each point represents a pair of base learners. Lower Cohen's κ indicates weaker agreement and greater prediction diversity; the vertical axis shows pairwise mean link-prediction test loss.

AI USE STATEMENT

Generative AI tools were used to assist code migration, debugging, experiment organization, and language editing. All experimental outputs reported in the paper were produced by executed code and were checked against the saved result files. The authors reviewed the AI-assisted code and prose and take responsibility for the final methods, claims, and artifacts.

ETHICS STATEMENT

This work introduces a graph-learning method and reuses previously released benchmark data. We did not collect new human-subject data. The method may inherit sampling and labeling biases present in the underlying path datasets; applications in consequential domains should therefore validate both the data provenance and the behavior of the learned state representation.

REPRODUCIBILITY STATEMENT

The project records the graph-construction parameters, fixed cross-validation split, random seeds, per-fold metrics, and per-learner probability outputs. The released artifact will include the scripts for trigram recovery, concise graph construction, graph validation, DGE-bag training, and probability-level ensembling, together with commands for the configurations reported in the tables. The public DGE and concise-network repositories are cited below.

REFERENCES

Semih Canturk, Renming Liu, Olivier Lapointe-Gagne, Vincent Letourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampasek. Graph positional and structural encoder. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5533–5566. PMLR, 2024. URL https://proceedings.mlr.press/v235/canturk24a.html.

Chris H. Q. Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

Moshe Eliasof, Eldad Haber, and Eran Treister. pathgcn: Learning general graph spatial operators from paths. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5878–5891, 2022. URL https://proceedings.mlr.press/v162/eliasof22a.html.

Moshe Eliasof, Fabrizio Frasca, Beatrice Bevilacqua, Eran Treister, Gal Chechik, and Haggai Maron. Graph positional encoding via random feature propagation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 9202–9223. PMLR, 2023. URL https://proceedings.mlr.press/v202/eliasof23a.html.

Yu-De Fu, Xiang Li, Ji-Chao Li, Mengjun Yu, Xiong-Yi Lu, Qizi Huangpeng, and Xiao-Jun Duan. Multi-scale higher-order dependencies (mshod): Higher-order interactions mining and key nodes identification for global liner shipping network. *Journal of Marine Science and Engineering*, 12 (8):1305, 2024. doi: 10.3390/jmse12081305.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Yiming Huang and Tolga Birdal. Hog-diff: Higher-order guided diffusion for graph generation. In *International Conference on Learning Representations*, 2026. URL https://circle-group.github.io/research/hog-diff/.

Di Jin, Rui Wang, Meng Ge, Dongxiao He, Xiang Li, Wei Lin, and Weixiong Zhang. Raw-gnn: Random walk aggregation based graph neural network. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2108–2114, 2022. doi: 10.24963/ijcai.2022/293.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Steven J. Krieg, Peter M. Kogge, and Nitesh V. Chawla. Growhon: A scalable algorithm for growing higher-order networks of sequences. In *Complex Networks & Their Applications IX*, pp. 485–496. Springer, 2021. doi: 10.1007/978-3-030-65351-4_39.

Steven J. Krieg, William C. Burgis, Patrick M. Soga, and Nitesh V. Chawla. Deep ensembles for graphs with higher-order dependencies. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Y1x4WTOmIn.

Jia-Xu Li, Xiaoqian Yuan, Yu-De Fu, Ji-Chao Li, Wenhui Tan, and Xin Lu. Representing significant dependencies with variable orders in networks. *Chaos, Solitons & Fractals*, 201:117405, 2025. doi: 10.1016/j.chaos.2025.117405.

Yuxuan Lin, Hanjing Yan, Xuan Tong, Yang Chang, Huanzhen Wang, Ziheng Zhou, Shuyong Gao, Yan Wang, and Wenqiang Zhang. Commonality in few: Few-shot multimodal anomaly detection via hypergraph-enhanced memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 7015–7023, 2026. doi: 10.1609/aaai.v40i9.37636.

Martin Rosvall, Alcides V. Esquivel, Andrea Lancichinetti, Jevin D. West, and Renaud Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications*, 5:4630, 2014.

Mandana Saebi, Jian Xu, Lance M. Kaplan, Bruno Ribeiro, and Nitesh V. Chawla. Efficient modeling of higher-order dependencies in networks: From algorithm to application for anomaly detection. *EPJ Data Science*, 9:15, 2020. doi: 10.1140/epjds/s13688-020-00233-y.

Rohit Sahasrabuddhe, Renaud Lambiotte, and Martin Rosvall. Concise network models of memory dynamics reveal explainable patterns in path data. *Science Advances*, 11:eadw4544, 2025. doi: 10.1126/sciadv.adw4544.

Jing Xiao, Yu-Cheng Zou, and Xiao-Ke Xu. Higher-order community detection by motif-based modularity optimization. *IEEE Transactions on Big Data*, 11:2529–2544, 2025. doi: 10.1109/TBDATA.2025.3544129.

Jian Xu, Thanuka L. Wickramarathne, and Nitesh V. Chawla. Representing higher-order dependencies in networks. *Science Advances*, 2(5):e1600028, 2016. doi: 10.1126/sciadv.1600028.

Tuo Xu and Lei Zou. Rethinking higher-order representation learning with graph neural networks. In *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pp. 38:1–38:25. PMLR, 2024. URL `https://proceedings.mlr.press/v231/xu24a.html`.

Yanxin Zhang, Yude Fu, Jichao Li, Peng Liu, and Jiaxu Li. Complex-metric higher-order dependency network with adaptive backoff for global shipping trajectory modeling. In *2026 12th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 1021–1028, 2026. doi: 10.1109/BigDIA71496.2026.11684635.