DUPLEX: Dual GAT for Complex Embedding of Directed Graphs

Zhaoru Ke [* 1 2] **Hang Yu** [* 2] **Jianguo Li** [2] **Haipeng Zhang** [1]

Abstract

Current directed graph embedding methods build upon undirected techniques but often inadequately capture directed edge information, leading to challenges such as: (1) **Suboptimal representations for nodes with low in/out-degrees**, due to the insufficient neighbor interactions; (2) **Limited inductive ability** for representing new nodes post-training; (3) **Narrow generalizability**, as training is overly coupled with specific tasks. In response, we propose DUPLEX, an inductive framework for complex embeddings of directed graphs. It (1) leverages Hermitian adjacency matrix decomposition for comprehensive neighbor integration, (2) employs a dual GAT encoder for directional neighbor modeling, and (3) features two parameter-free decoders to decouple training from particular tasks. DUPLEX outperforms state-of-the-art models, especially for nodes with sparse connectivity, and demonstrates robust inductive capability and adaptability across various tasks. The code is available at https://github.com/alipay/DUPLEX.

1. Introduction

Graphs, as a powerful and versatile data structure, have cemented their importance across a myriad of domains, including social science (Hu et al., 2017), recommendation systems (Wu et al., 2022), bioinformatics (Yue et al., 2020), traffic prediction (Zheng et al., 2020), financial and risk analysis (Yang et al., 2021; Yu et al., 2022). At the heart of graph analytics lies the concept of graph embedding, which seeks to encode complex, high-dimensional graph structures into compact, low-dimensional vector spaces. These repre-

*Equal contribution. This work was done when Zhaoru Ke was a research intern at Ant Group. [1]School of Information Science and Technology, ShanghaiTech University, China [2]Ant Group, China. Correspondence to: Haipeng Zhang <zhanghp@shanghaitech.edu.cn>, Jianguo Li <lijg.zero@antgroup.com>.
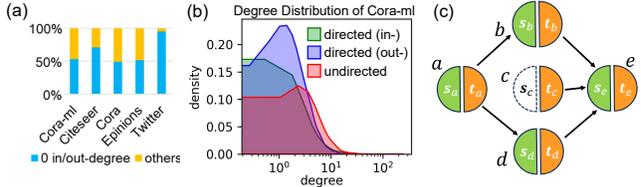
Figure 1. (a) Many nodes in real digraphs have zero in/out-degree. (b) Separating in/out-degree yields more low-degree nodes compared to the total degree that disregards direction. (c) Lack of in-neighbors hinders the dual embedding methods in capturing node c's source role.

sentations are then applied to a variety of predictive tasks, such as link prediction and node classification. While the research on graph embeddings has achieved notable success, the primary focus has been on undirected graphs, which often fail to capture the intricate directional relationships inherent in many real-world networks.

Take, for example, the realm of social media, where users (nodes) follow one another, creating a digraph (directed graph) of influence and information flow. In this scenario, a directed edge from user u to user v implies that user u follows user v, but not necessarily vice versa. The directionality of these edges encodes significant information about user behavior, influence patterns, and community structure. The need for digraph embedding (DGE) methods that can proficiently encode such asymmetrical structures is evident, yet the development of an effective DGE framework must also navigate several key desiderata to ensure practicality and utility:

d1. **Performance Across Various Node Degrees:** Social networks are characterized by a wide disparity in user connectivity. An efficacious DGE method should consistently represent both influential personalities and average individuals, irrespective of the network's sparsity. This challenge is more pronounced in digraphs, where the separation of in and out neighbors often results in a higher proportion of nodes with a skewed degree distribution compared to their undirected counterparts (illustrated in Figure 1(a)-(b)).

d2. **Transductive and Inductive Learning:** The fluidity of social networks, marked by the continual arrival and departure of users, demands a DGE method capable of both updating embeddings for current users (transductive learning) and seamlessly extending to new users (inductive

Table 1. Existing DGE methods and their specifics.

	Embedding	Network	Learning Algo.
HOPE (Ou et al., 2016), APP (Zhou et al., 2017), NERD (Khosla et al., 2019), ODIN (Yoo et al., 2023), DGGAN (Zhu et al., 2020)	dual	shallow (no network)	self-supervised
DiGAE (Kollias et al., 2022)	dual	spectral GNN	self-supervised
BLADE (Virinchi & Saladi, 2023), CoBA (Liu et al., 2023)	dual	spatial GNN	self-supervised
Gravity GAE (Salha-Galvan et al., 2019)	dual	spectral GNN	self-supervised
Dhypr (Zhou et al., 2021)	dual	spatial GNN	self-supervised
MagNet (Zhang et al., 2021), Framelet-MagNet (Lin & Gao, 2023), SigMaNet (Fiorini et al., 2022)	single	spectral GNN	supervised
UGCL (Ko et al., 2023)	single	spectral GNN	self-supervised
DUPLEX (proposed method)	single	spatial GNN	self-supervised

learning), without re-training from scratch.

***d3*. Task Agnosticism:** The embeddings produced by the DGE method should lend themselves to a variety of tasks such as detecting communities, recommending new connections, and identifying influential users, without being tailored to any single application.

Regrettably, our survey of the existing landscape reveals a gap: **existing DGE methods fall short in simultaneously satisfying all these desiderata**. As evidenced in Table 1, while techniques that rely on self-supervised spatial GNNs (graph neural networks) (Ou et al., 2016; Zhou et al., 2017; Khosla et al., 2019; Yoo et al., 2023; Kollias et al., 2022; Virinchi & Saladi, 2023) demonstrate robustness against inductive bias and different downstream tasks, they typically learn dual embeddings (source and target roles) for each node, so as to better represent edge asymmetries. Consequently, they are prone to inferior performance for low in/out-degree nodes, since it fails to update the source embedding for low-out-degree nodes and the target embedding for low-in-degree nodes due to insufficient neighbors for training (cf. Figure 1(c)). On the flip side, single-embedding strategies (Zhang et al., 2021; Lin & Gao, 2023; Fiorini et al., 2022; Ko et al., 2023) can alleviate this issue by aggregating information from both in and out neighbors. However, they are generally bound to spectral GNN architectures and fully supervised paradigms, limiting their scope to specific tasks within a transductive context.

To bridge the above gap, we propose **DUPLEX** (**DU**al graph attention networks for com**PLEX** embedding of digraphs), a novel approach for digraph embedding. Our method leverages a dual graph attention network (GAT) encoder and two parameter-free decoders to learn a single complex embedding for each node. Specifically, to tackle the problem for those low-degree nodes (***d1***), DUPLEX embraces complex embeddings, underpinning the source and target roles as complex conjugates, which facilitates a collaborative op-

timization leveraging both in and out neighbors. For the inductive power (***d2***), DUPLEX employs a dual GAT encoder with separate components for the amplitude and phase parts of the complex node embedding. The amplitude encoder captures connection information with an undirected graph aggregator, while the phase encoder characterizes direction information using a digraph aggregator. Both aggregators update node embeddings by collecting information only from neighboring nodes, eliminating the need to access the entire graph and generalizing well to unseen nodes. Lastly, to achieve task generalization (***d3***), DUPLEX employs the two parameter-free decoders (direction-aware and connection-aware) for reconstructing the Hermitian adjacency matrix (HAM) of the digraph. The node embeddings are then trained in a self-supervised manner, preserving the structural characteristics. Thus, the resulting node embedding can adapt effectively to various downstream tasks. Our contributions can be summarized as:

- We propose DUPLEX that learns **complex** node embeddings. Our approach incorporates a **dual GAT encoder**, comprising two specially designed graph aggregators for the amplitude and phase components. To our knowledge, this is the first exploration of using spatial GNNs for complex embeddings of digraphs.

- We propose two parameter-free decoders specially designed for complex embeddings, targeting at the reconstruction of the HAM of the digraph. The model can be trained in a self-supervised manner, enabling the adaptability of node embeddings across various tasks.

- We conduct comprehensive experiments on diverse tasks and datasets, showcasing the superior performance of our approach. The results demonstrate its efficacy in modeling low-degree nodes, generalizing to multiple tasks, and handling unseen nodes.

2. Related Works

The following discussion reviews existing digraph embedding (DGE) methodologies from three perspectives: the nature of the embeddings (dual vs. single), the network designs (transductive vs. inductive), and the learning paradigms (supervised vs. self-supervised). These insights further underscore the innovative design of DUPLEX.

2.1. Dual versus Single Node Embeddings

As shown in the second column of Table 1, studies on DGE have largely diverged into two camps based on how they treat node roles: those that generate dual embeddings to capture distinct source and target roles and those that distill this information into a single embedding. Pioneering approaches like APP (Zhou et al., 2017) often fall short for sparsely connected nodes due to inadequate sampling of these nodes within random walks. Although improved

sampling strategies from NERD (Khosla et al., 2019) and BLADE (Virinchi & Saladi, 2023) aim to remedy this, their effectiveness is curtailed by the small neighborhood sets of low-degree nodes, which are insufficient for accurate embeddings. DGGAN (Zhu et al., 2020) and CoBA (Liu et al., 2023) strive to settle this problem by bolstering the link between source and target embeddings, but only in a heuristic way. Alternatively, GravityGAE (Salha-Galvan et al., 2019) and its hyperbolic extension (Zhou et al., 2021) employs a gravitational analogy with a mass parameter only for the target role, yet struggles with nodes of meager in-degrees.

Conversely, single-embedding methods, exemplified by MagNet (Zhang et al., 2021) and its extensions (Lin & Gao, 2023; Fiorini et al., 2022; Ko et al., 2023), effectively consolidate information from both in and out neighbors into a unified complex embedding, overcoming the embedding quality challenges for nodes with low in/out-degrees.

DUPLEX aligns with the latter approach, using the HAM akin to the magnetic Laplacian. Yet, it advances the notion by modeling the source and target roles as complex conjugates, enabling joint optimization while preserving edge asymmetries.

2.2. Transductive versus Inductive Network Designs

As shown in the third column of Table 1, the body of DGE work can also be classified based on its adaptability to unseen data. Shallow methods such as HOPE (Ou et al., 2016) and APP (Zhou et al., 2017), which adapt classical techniques like matrix factorization and random walks to digraphs, operate under transductive settings where embeddings are directly learned as trainable parameters. Spectral GNNs, including DiGAE (Kollias et al., 2022), Gravity GAE (Salha-Galvan et al., 2019), and MagNet (Zhang et al., 2021), further necessitate full graph knowledge during graph Fourier transform, limiting their applicability to new nodes. Spatial GNNs, represented by BLADE (Virinchi & Saladi, 2023), CoBA (Liu et al., 2023), and Dhypr (Zhou et al., 2021), however, infer embeddings by aggregating local neighborhood information, enabling them to generalize to nodes absent during training.

DUPLEX builds upon this spatial approach with a dual encoder architecture that separately tackles the amplitude and phase aspects of embeddings, thereby accommodating both transductive and inductive learning effectively.

2.3. Supervised versus Self-supervised Learning

As shown in the fourth column of Table 1, supervised methods such as MagNet (Ou et al., 2016) and subsequent variations (Lin & Gao, 2023; Fiorini et al., 2022; Ko et al., 2023) leverage spectral GNN encoders with linear or convolutional decoders, trained end-to-end for specific tasks.

This task-specific focus can hinder the generalizability of node embeddings, as they may not fully capture the graph's structural features. On the other hand, self-supervised methods (Ou et al., 2016; Yoo et al., 2023; Liu et al., 2023) differ by encoding both connectivity and directionality within the graph, producing embeddings that are suitable for various downstream tasks.

As a result, DUPLEX employs a self-supervised method and reconstructs the Hermitian adjacency matrix through two parameter-free decoders, such that the embeddings given by the learnable encoder not only align with the graph's structural properties but also ensure the adaptability of the embeddings across multiple tasks.

3. Method

A digraph (directed graph) $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined by its nodes $\mathcal{V}$ and directed edges $\mathcal{E}$, with each edge $(u, v) \in \mathcal{E}$ representing a connection from node u to node v. Our goal is to represent each node u by a d-dimensional complex vector $\boldsymbol{x}_u \in \mathbb{C}^{d \times 1}$ that meets the desiderata presented in Section 1. To this end, we introduce DUPLEX, a novel framework that is illustrated in Figure 2(a). At the heart of DUPLEX lies the Hermitian adjacency matrix (HAM), whose symmetrical structure and complex-valued representation enable the integration of both directionality and connectivity within a cohesive framework. This mathematical formalism guides the design of DUPLEX's embeddings, encoder, decoder, and loss functions, ensuring coherence with HAM's fundamental principles. Specifically, beginning with initial embeddings[1], the dual GAT encoder generates amplitude and phase embeddings that encode the graph's connectivity and directionality, respectively. A fusion layer within the encoder integrates these embeddings, while subsequent direction-aware and connection-aware decoders—both parameter-free—rebuild the HAM. This architecture ensures that the learned complex embeddings authentically represent the graph's structure, guided by self-supervised loss functions.

In the sequel, we introduce each component in DUPLEX, including the HAM and the resulting embedding formulations, the dual GAT encoder, the decoders, and the loss functions.

3.1. Hermitian Adjacency Matrix Construction

Recall that classical undirected graph embedding methods often employ matrix factorization techniques that can be represented as $\boldsymbol{A} = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}$, where $\boldsymbol{A}$ is the symmetric adjacency matrix and $\boldsymbol{X}$ is the corresponding real-valued node embedding matrix. In contrast, digraphs feature asymmetric adjacency matrices, resulting in factorizations of the form $\boldsymbol{A} = \boldsymbol{X}_s^{\mathsf{T}}\boldsymbol{X}_t$, with $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ denoting the source and tar-

---

[1]For non-attributed graphs, we use the standard normal distribution for initialization.
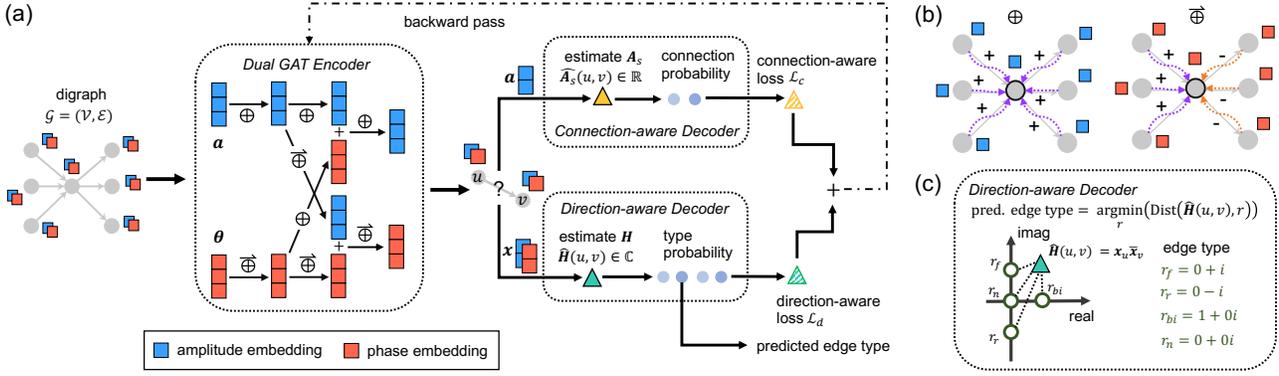
Figure 2. The architecture of DUPLEX. (a) Forward pass and backward pass of the model. (b) The undirected ($\oplus$) and directed ($\vec{\oplus}$) graph aggregator. (c) The main idea of the direction-aware decoder.

get embeddings, respectively. As highlighted in Section 2.1, such dual embedding strategies encounter difficulties in generating high-quality embeddings for nodes with low in/out-degree. More discussions from the perspective of SVD (singular value decomposition) are provided in Appendix B.

To address these limitations, we adopt the Hermitian adjacency matrix (HAM), a symmetric matrix utilized within spectral graph theory (Guo & Mohar, 2015; Liu & Li, 2015) to represent digraphs. The HAM $\boldsymbol{H}$ for a digraph $\mathcal{G}$ is defined in polar form as:

$$\boldsymbol{H} = \boldsymbol{A}_s \odot \exp\left(i\frac{\pi}{2}\boldsymbol{\Theta}\right), \tag{1}$$

where i represents the imaginary unit, π is the known mathematical constant, and $\odot$ signifies the Hadamard product. The symmetric binary matrix $\boldsymbol{A}_s$ satisfies $\boldsymbol{A}_s(u,v) = 1$ if $(u,v) \in \mathcal{E} \vee (v,u) \in \mathcal{E}$, and 0 otherwise. The antisymmetric matrix $\boldsymbol{\Theta}$ contains elements from the set $\{-1, 0, 1\}$, as defined by:

$$\boldsymbol{\Theta}(u,v) = \begin{cases} 1, & \text{if } (u,v) \in \mathcal{E}, \\ -1, & \text{if } (v,u) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

As a result, $\boldsymbol{H}(u,v) \in \{i, -i, 1, 0\}$ alone can represent forward, reverse, bidirectional, and no edge between u and v, whereas the asymmetric adjacency matrix $\boldsymbol{A}$ requires both $\boldsymbol{A}(u,v)$ and $\boldsymbol{A}(v,u) \in \{0,1\}$ for the same goal.

Moreover, the matrix decomposition $\boldsymbol{H} = \boldsymbol{X}^\intercal \bar{\boldsymbol{X}}$ allows for the derivation of the node embedding $\boldsymbol{x}_u$ in polar form:

$$\boldsymbol{x}_u = \boldsymbol{a}_u \odot \exp\left(i\frac{\pi}{2}\boldsymbol{\theta}_u\right), \tag{3}$$

$$\bar{\boldsymbol{x}}_u = \boldsymbol{a}_u \odot \exp\left(-i\frac{\pi}{2}\boldsymbol{\theta}_u\right). \tag{4}$$

Here, $\boldsymbol{a}_u$ encapsulates the amplitude and $\boldsymbol{\theta}_u$ the phase of $\boldsymbol{x}_u$. We can interpret $\boldsymbol{x}_u$ and $\bar{\boldsymbol{x}}_u$ as the complex conjugate embeddings representing the source and target roles of node u,

in contrast to the independent dual embeddings of previous methods. This embedding representation is a joint function of $\boldsymbol{a}_u$ and $\boldsymbol{\theta}_u$, facilitating co-optimized learning from both in and out-neighborhoods, which will be explained in Section 3.2, thus resolving the issue of suboptimal embeddings for nodes with low degree connectivity. From another perspective, $\boldsymbol{x}_u$ can be considered as a unified embedding vector for node u, with the HAM reconstructible through the Hermitian inner product between node embeddings instead of the ordinary inner product.

3.2. Dual GAT Encoder

The encoder in DUPLEX aims to generate the amplitude $\boldsymbol{a}_u$ and phase $\boldsymbol{\theta}_u$ embeddings, which combine to form the complex node embeddings $\boldsymbol{x}_u$. One of the nice properties of the HAM is the separation of the amplitude $\boldsymbol{A}_s$ and phase $\boldsymbol{\Theta}$ components to capture the connection and direction information, as shown in Eq. (1). Moreover, Eqs. (3) and (4) reveal that the amplitude parts of the source and target embeddings are identical for a node, but the phase parts exhibit opposite signs. In other words, the phase embedding is direction-aware, whereas the amplitude embedding is not. This insight lays the groundwork for a dual encoder that separately processes amplitude and phase, enabling streamlined learning and halving the time complexity and parameter count compared to direct complex embeddings encoding (detailed in Appendix D).

Our dual encoder comprises an amplitude encoder, a phase encoder, and a fusion layer, all based on the spatial GAT backbone for its flexibility and scalability (Velickovic et al., 2018). Note that GAT can be replaced by other spatial GNNs in DUPLEX (see Section 4.3). Both encoders update embeddings by aggregating information from in and out neighbors, mitigating the issue of poor embeddings for low-degree nodes, as we further discuss now.

3.2.1. AMPLITUDE ENCODER

Since the amplitude embedding $\boldsymbol{a}_u$ captures connection information akin to undirected graphs, we utilize the original GAT (Velickovic et al., 2018) to update $\boldsymbol{a}_u$:

$$\boldsymbol{a}'_u = \phi\Big(\oplus \big(\{\boldsymbol{a}_v, \forall v \in \mathcal{N}(u)\}\big)\Big), \qquad (5)$$

where $\boldsymbol{a}'_u$ denotes the updated amplitude embedding, $\mathcal{N}(u)$ includes node u and its neighbors, ϕ is the activation function (ReLU in our case), and $\oplus(\cdot)$ is the aggregator used for the undirected graph (see Figure 2(b)):

$$\oplus\big(\{\boldsymbol{a}_v, \forall v \in \mathcal{N}(u)\}\big) = \sum_{v \in \mathcal{N}(u)} f_a(\boldsymbol{a}_u, \boldsymbol{a}_v)\psi_a(\boldsymbol{a}_v). \quad (6)$$

Here, $f_a(\boldsymbol{a}_u, \boldsymbol{a}_v)\psi_a(\boldsymbol{a}_v)$ is a learnable attention mechanism that computes the relevance of neighbor embeddings to node u.

3.2.2. PHASE ENCODER

Bridging the gap between undirected graph embedding approaches and the directionality of digraphs, the phase encoder introduces a novel, direction-aware graph aggregator. This aggregator respects the sign difference of $\boldsymbol{\theta}_u$ in Eqs. (3) and (4). Specifically, it consumes information from the in-neighbors and out-neighbors with different signs (as depicted in Figure 2(b)), enabling the phase encoder to effectively incorporate the contributions from both types of neighbors while considering their inherent asymmetry:

$$\vec{\oplus}\big(\{\boldsymbol{\theta}_v : v \in \mathcal{N}(u)\}\big) = \sum_{v \in \mathcal{N}_{\text{in}}(u)} f_\theta(\boldsymbol{\theta}_u, \boldsymbol{\theta}_v)\psi_\theta(\boldsymbol{\theta}_v)$$
$$- \sum_{v \in \mathcal{N}_{\text{out}}(u)} f_\theta(\boldsymbol{\theta}_u, \boldsymbol{\theta}_v)\psi_\theta(\boldsymbol{\theta}_v), \qquad (7)$$

where $\mathcal{N}_{\text{in}}(u)$ and $\mathcal{N}_{\text{out}}(u)$ denote the sets of in-neighbors and out-neighbors, respectively. The resulting phase embedding in each layer can be updated as:

$$\boldsymbol{\theta}'_u = \phi\Big(\vec{\oplus}\big(\{\boldsymbol{\theta}_v : v \in \mathcal{N}(u)\}\big)\Big). \qquad (8)$$

This design choice results in a phase encoder that is distinctly directional, setting it apart as a novel contribution to graph representation learning.

3.2.3. FUSION LAYER

The fusion layer functions as a pivotal junction where the amplitude and phase embeddings intersect, each carrying unique yet complementary information essential for the reconstruction of the HAM. Let us consider the amplitude encoder as an illustrative example. In the fusion layer, we not only aggregate the information from the amplitude embeddings but also gather side information from the phase

embeddings of the previous layer using the undirected graph aggregator $\oplus$. Mathematically, this can be formulated as:

$$\boldsymbol{a}'_u = \phi\Big(\oplus \big(\{\boldsymbol{a}_v, \forall v \in \mathcal{N}(u)\}\big) + \oplus\big(\{\boldsymbol{\theta}_v, \forall v \in \mathcal{N}(u)\}\big)\Big). \qquad (9)$$

The resulting $\boldsymbol{a}'_u$ is then propagated to the subsequent layers in the amplitude encoder. On the other hand, for the phase encoder, the fusion layer can be similarly expressed by replacing $\oplus$ with $\vec{\oplus}$ in Eq. (9).

Our approach employs a "mid-fusion" strategy, integrating embeddings at the network's intermediate layers for two main reasons. First, in the absence of attributes, node features are initially random, and early fusion might introduce noise rather than beneficial information. Second, fusing at the terminal layer could dilute the unique attributes of amplitude and phase embeddings that encode connection and directional information, respectively, potentially hindering the decoders' ability to accurately reconstruct the HAM. Consequently, the fusion layer is optimally placed as an intermediary, allowing for the coordinated optimization within the dual encoder framework.

The integration of the fusion layer within the DUPLEX framework is also founded on the hypothesis that an adequately flexible encoder can achieve superior performance by permitting the controlled interchange of information between the amplitude and phase. This fusion is implemented utilizing the attention mechanism, where the resulting attention score dictates the strength of the interchange. A critical aspect of this implementation is that when the attention score is minimal, nearing zero, it effectively precludes the exchange of information between the amplitude and phase. Consequently, the data itself, from which the attention scores are adaptively learned, governs the presence and extent of information exchange.

Relation to GAT: DUPLEX amounts to GAT for undirected graphs after removing the phase or direction-related components. In other words, DUPLEX is an extension of GAT that is augmented for digraphs through the integration of direction information, and so inherits the merits of GAT.

3.3. Parameter-free Decoder and Self-supervised Loss

After obtaining the complex embeddings for nodes, DUPLEX proceeds to reconstruct the HAM with two parameter-free decoders (i.e., direction and connection-aware decoders), each paired with its own self-supervised loss function. Unlike previous methods in Table 1 that reconstruct the real-valued asymmetric adjacency matrices with the connection-aware decoder, the direction-aware decoder in DUPLEX is specially designed for the complex-valued HAM. Meanwhile, the connection-aware decoder complements the former by focusing on the binary existence of connections, effectively serving as an auxiliary to the direction-

aware objective. The parameter-free decoders and self-supervised losses enable DUPLEX to learn both connectivity and directionality within the digraph, fostering node embeddings that generalize effectively across tasks without depending on tailored parameterized decoders.

3.3.1. DIRECTION-AWARE DECODER AND LOSS

Recall that HAM comprises four distinct elements, namely $\boldsymbol{H}(u,v) \in \mathcal{R} = \{i, -i, 1, 0\}$, which respectively signify forward, reverse, bidirectional, and no edges between any pair of nodes u and v. The reconstruction task models the estimated matrix elements $\hat{\boldsymbol{H}}(u,v) = \boldsymbol{x}_u^{\mathsf{T}} \bar{\boldsymbol{x}}_v$ to align with the ground truth $\boldsymbol{H}(u,v)$. Recognizing the limitations of low-rank embeddings in capturing the full spectrum of the HAM, we approach the problem as a classification task. Each edge (u,v) is assigned probabilities corresponding to its edge type, based on the relative distance between $\hat{\boldsymbol{H}}(u,v)$ and $\boldsymbol{H}(u,v)$ (illustrated in Figure 2(c)):

$$P(\hat{\boldsymbol{H}}(u,v) = r) = \frac{\exp(-|\boldsymbol{x}_u^{\mathsf{T}} \bar{\boldsymbol{x}}_v - r|)}{\sum_{r' \in \mathcal{R}} \exp(-|\boldsymbol{x}_u^{\mathsf{T}} \bar{\boldsymbol{x}}_v - r'|)}, \quad \forall r \in \mathcal{R}, \tag{10}$$

where the L1 distance is used, due to its advantage over L2 (Appendix J.4). We then minimize the negative log-likelihood of the samples pertaining to different edge types, resulting in the self-supervised direction-aware loss:

$$\mathcal{L}_d = -\sum_{r \in \mathcal{R}} \sum_{\boldsymbol{H}(u,v)=r} \log P(\hat{\boldsymbol{H}}(u,v) = r). \tag{11}$$

3.3.2. CONNECTION-AWARE DECODER AND LOSS

The connection-aware decoder isolates the task of discerning node connectivity by reconstructing $\boldsymbol{A}_s$, the amplitude component of the HAM (1), from the amplitude embeddings $\boldsymbol{a}_u$. It posits the connection probability for an edge (u,v) as:

$$P(\hat{\boldsymbol{A}}_s(u,v) = 1) = \sigma(\boldsymbol{a}_u^{\mathsf{T}} \boldsymbol{a}_v), \tag{12}$$

where σ is the sigmoid function and $\hat{\boldsymbol{A}}_s$ is the estimated connection matrix. This decoder's loss function $\mathcal{L}_c$ adheres to the same negative log-likelihood minimization principle as the direction-aware loss.

3.3.3. TOTAL LOSS

The total loss can be written as: $\mathcal{L} = \mathcal{L}_d + \lambda \mathcal{L}_c$, where λ is the weight parameter for the connection-aware loss $\mathcal{L}_c$. The optimization target of the connection-aware loss is subsumed within the broader objective of the direction-aware loss. In tandem, the connection-aware loss can support the direction-aware loss by constraining the optimization space in the initial stages. As a result, we start with a predefined value of $\lambda = \lambda_0$, and allow λ to decay (i.e., $\lambda = \lambda_0 q^k$) with a decay factor $q < 1$ as the number of epochs k increases,

Table 2. Dataset characteristics.

Dataset	$|\mathcal{V}|$	$|\mathcal{E}|$	%Directed Edges	Feature Dim	#Classes
Cora-ml	2,995	8,416	93.9	2,879	7
Citeseer	3,312	4,715	95.0	3,703	6
Cora	23,166	91,500	94.9	-	-
Epinions	75,879	508,837	59.5	-	-
Twitter	465,017	834,797	99.7	-	-

reflecting the decreasing necessity of the connection-aware loss as direction-aware accuracy improves. This objective function implicitly mandates $\boldsymbol{a}_u$ and $\boldsymbol{\theta}_u$ to encapsulate the amplitude and phase information respectively, given the encoders possess sufficient flexibility to facilitate such characterization.

3.3.4. SUPERVISED TRAINING OF DUPLEX

Apart from self-supervised training, DUPLEX can be readily trained end-to-end in a supervised manner for specific downstream tasks. This entails replacing the decoder and loss function with a task-specific objective. Taking node classification as an example, given the complex embeddings encoded by the dual GAT encoder, we can simply concatenate the amplitude and phase embeddings. These concatenated embeddings are then mapped to node labels using a linear layer as in (Zhang et al., 2021). The model is then trained using a cross-entropy loss function, optimizing its performance specifically for the node classification task.

Relation to the MagNet series: MagNet (Zhang et al., 2021) and its variants (Fiorini et al., 2022; Lin & Gao, 2023) mainly extend spectral GCNs from undirected to digraphs by substituting the traditional Laplacian in graph convolutions with its magnetic counterpart, combining amplitude and phase embeddings, and utilizing a linear decoder for link prediction tasks. While both the magnet Laplacian and the HAM are Hermitian matrices, MagNet's approach is rooted in spectral domain convolutions. In contrast, DUPLEX innovatively employs spatial GNNs derived from the properties of HAM, with the goal of further reconstructing the HAM. Furthermore, unlike the MagNet series, which predominantly operates under fully supervised training, DUPLEX takes advantage of a self-supervised learning paradigm.

4. Experiments

In this section, we benchmark DUPELX against existing digraph embedding methods on three tasks, namely, link prediction, transductive node classification, and inductive node classification. Furthermore, we conduct a comprehensive ablation study on DUPLEX to assess the significance of its different components.

Table 3. Link prediction AUC (%) and ACC (%) for subtask 1 and 2. The best results are in **bold** and the second are <u>underlined</u>.

Method	Existence Prediction								Direction Prediction							
	Citeseer		Cora		Epinions		Twitter		Citeseer		Cora		Epinions		Twitter	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
NERD	80.4(0.6)	79.0(1.1)	85.1(0.5)	72.8(0.1)	78.6(0.1)	66.5(0.1)	94.9(0.0)	77.6(0.1)	82.1(1.0)	83.1(0.6)	92.3(0.2)	76.2(0.5)	86.9(0.3)	55.6(0.1)	95.6(0.1)	80.9(0.1)
DGGAN	84.5(0.9)	50.1(0.0)	89.6(0.2)	54.6(3.0)	84.5(2.7)	50.0(0.0)	99.1(0.1)	77.8(8.1)	89.4(0.8)	50.0(0.0)	95.4(0.2)	57.1(5.0)	94.0(2.0)	50.0(0.0)	99.1(0.1)	80.7(13)
ODIN	86.2(1.1)	76.8(1.5)	90.8(0.2)	83.1(0.2)	<u>90.9(0.1)</u>	83.3(0.2)	<u>99.2(0.0)</u>	**98.7(0.1)**	95.4(0.8)	87.5(1.2)	96.7(0.2)	90.5(0.2)	**97.4(0.0)**	92.1(0.1)	99.8(0.0)	99.6(0.0)
DiGAE	87.1(1.9)	72.8(2.1)	83.9(0.5)	70.1(0.2)	81.8(0.1)	70.2(0.1)	98.8(0.0)	56.7(0.0)	78.5(2.1)	56.8(1.7)	89.8(0.3)	71.2(0.2)	91.5(0.1)	55.6(0.1)	**99.9(0.0)**	50.0(0.0)
MagNet	88.3(0.4)	80.7(0.8)	89.4(0.1)	81.4(0.3)	85.1(0.1)	77.5(0.3)	99.1(0.1)	97.7(0.1)	96.4(0.6)	91.7(0.9)	95.4(0.2)	88.9(0.4)	96.6(0.1)	92.1(0.0)	<u>99.9(0.1)</u>	98.5(0.9)
SigMaNet	91.4(0.8)	84.4(3.2)	93.6(0.2)	87.7(0.4)	90.3(0.0)	82.5(0.0)	99.1(0.0)	<u>98.3(0.0)</u>	98.3(0.3)	97.8(0.9)	96.4(0.0)	94.7(0.1)	<u>96.7(0.0)</u>	92.5(0.0)	**99.9(0.0)**	99.7(0.0)
DUPLEX*	98.0(0.7)	95.3(0.5)	95.8(0.3)	93.1(0.1)	90.7(0.2)	<u>84.5(0.5)</u>	96.1(0.6)	96.9(0.2)	99.3(0.5)	98.3(0.3)	97.1(0.2)	95.5(0.3)	93.9(0.2)	92.2(0.0)	**99.9(0.0)**	<u>99.7(0.0)</u>
DUPLEX	**98.6(0.4)**	**95.7(0.5)**	**95.9(0.1)**	**93.2(0.1)**	**91.0(0.2)**	**85.5(0.0)**	**99.3(0.2)**	**98.7(0.1)**	**99.7(0.2)**	**98.7(0.4)**	**97.2(0.2)**	**95.9(0.1)**	95.2(0.4)	**92.6(0.1)**	**99.9(0.0)**	**99.8(0.0)**

Table 4. Link prediction ACC (%) for subtask 3 and 4.

Method	Three-type Classification				Four-type Classification			
	Citeseer	Cora	Epinions	Twitter	Citeseer	Cora	Epinions	Twitter
NERD	68.8(0.7)	67.7(0.6)	66.6(0.3)	73.2(0.1)	31.4(0.6)	38.6(0.8)	33.1(0.4)	32.2(0.4)
DGGAN	59.0(0.3)	58.8(0.1)	57.6(0.3)	70.0(5.0)	13.4(4.0)	10.5(6.4)	20.1(0.1)	67.4(7.1)
ODIN	67.2(0.8)	72.1(0.3)	87.3(0.1)	98.5(0.0)	67.1(0.8)	70.6(0.4)	73.1(0.1)	<u>98.0(0.0)</u>
DiGAE	83.7(1.3)	68.5(0.3)	80.4(0.2)	69.8(0.0)	42.3(0.6)	34.4(0.3)	40.2(0.2)	35.9(0.0)
MagNet	72.0(0.9)	66.8(0.3)	76.9(0.9)	93.9(2.6)	69.3(0.4)	63.0(0.3)	65.2(0.4)	91.6(1.1)
SigMaNet	81.3(0.4)	80.3(0.2)	86.4(0.1)	98.0(0.0)	76.2(2.1)	78.7(0.4)	<u>75.4(0.1)</u>	97.2(0.0)
DUPLEX*	93.7(0.9)	92.1(0.1)	88.9(0.1)	99.0(0.1)	90.7(0.8)	88.3(0.2)	74.9(0.6)	93.6(0.3)
DUPLEX	**94.8(0.2)**	**92.2(0.1)**	**88.9(0.0)**	**99.2(0.1)**	**91.1(1.0)**	**88.4(0.4)**	**76.4(0.2)**	**98.1(0.2)**

4.1. Datasets and Experiment Set-up

The experiments are conducted on five public datasets of digraphs, namely Cora-ml, Citeseer, Cora, Epinions, and Twitter (overview in Table 2 and more details in Appendix E). Note that only Cora-ml and Citeseer datasets are suitable for node classification tasks, as they include node labels and initial attributes. The graph datasets used in our study are unweighted and directed. We then compare DUPLEX with a selection of state-of-the-art (SOTA) node embedding models for digraphs, including both dual and single embedding methods. The first group consists of **NERD** (Khosla et al., 2019), **DGGAN** (Zhu et al., 2020), **DiGAE** (Kollias et al., 2022) and **ODIN** (Yoo et al., 2023). For the second group, we consider **MagNet** (Zhang et al., 2021) and **SigMaNet** (Fiorini et al., 2022). In addition, we examine the performance of a simplified version of our model, denoted as **DUPLEX***, which excludes the fusion layer. To ensure optimal performance, we perform hyperparameter selection for each method. Moreover, we repeat the experiments 10 times and report the average values, along with the standard deviation across the runs (surrounded by brackets). For implementation details, please see Appendix F.

4.2. Downstream Tasks

4.2.1. LINK PREDICTION

Embarking on our exploration with link prediction, we adopt the approach delineated by Zhang et al.(Zhang et al., 2021)

to randomly split the datasets into training, validation, and testing subsets, maintaining a ratio of 16:1:3. To thoroughly evaluate the model's proficiency in discerning varied edge types, we engage in four prevalent subtasks: **Existence Prediction** (EP)(Zhu et al., 2020), which determines the likelihood of an edge's presence; **Direction Prediction** (DP)(Zhu et al., 2020; Zhang et al., 2021), ascertaining the orientation of unidirectional edges; **Three-type Classification** (TP)(Zhang et al., 2021), categorizing edges as positive, reverse, or non-existent; and **Four-type Classification** (FP), discerning positive, reverse, bidirectional, or non-existent edges. We present additional details in Appendix G.

Notably, **DUPLEX**, alongside **NERD**, **DGGAN**, **DiGAE**, and **ODIN**, employs a self-supervised approach to regenerate the directed graph, enabling the direct application of learned embeddings across all subtasks. The task-specific **MagNet** and **SigMaNet**, by contrast, necessitate individual end-to-end training for each specific subtask. The outcomes, including Area Under Curve (AUC) for subtasks EP and DP, and accuracy (ACC) for all tasks, are listed in Tables 3-4.

DUPLEX shines in the EP and DP experiments, claiming the highest AUC in 7 out of 8 cases and dominating in ACC across all subtasks, showcasing its adeptness in capturing a spectrum of node relationships within digraphs. Even without the fusion layer, the ablated DUPLEX* variant secures top or near-top AUC or ACC scores in 12 out of 16 experiments. Almost all methods excel in DP, particularly on larger datasets, likely due to their design focus on edge
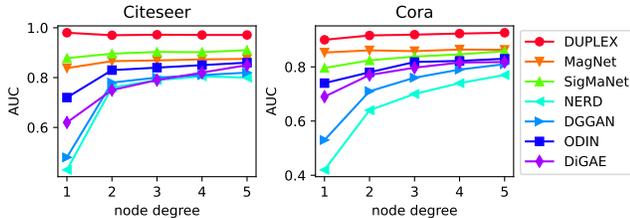
Figure 3. Link existence prediction AUC (%) under extremely low-degree setting.

Table 5. Transductive node classification results.

Method	Citeseer		Cora-ml	
	mac. F_1	mic. F_1	mac. F_1	mic. F_1
NERD	49.0(0.4)	52.5(0.7)	**78.8(2.3)**	**80.1(1.6)**
DGGAN	19.1(1.5)	22.7(1.9)	15.6(1.4)	23.6(3.7)
ODIN	17.7(2.5)	19.2(2.2)	12.6(2.1)	18.0(2.0)
DiGAE	26.6(3.1)	29.8(3.2)	55.1(4.8)	56.2(3.2)
DUPLEX-S	40.0(0.2)	42.6(1.1)	64.3(1.4)	65.7(1.2)
DUPLEX*	51.2(0.1)	54.3(2.0)	76.0(1.3)	77.8(1.6)
DUPLEX	**53.0(2.5)**	**56.2(1.7)**	77.9(0.6)	79.8(0.7)

directionality and the abundance of training data. Yet, DU-PLEX distinguishes itself in more nuanced tasks like TP and FP, and in smaller datasets. For instance, on the Citeseer dataset, DUPLEX records approximately 11.1% and 14.9% improvements in TP and FP, respectively. Additionally, ODIN delivers satisfying results across most tasks, which may stem from its strategic address of potential degree distribution discrepancies between training and testing sets. Despite this, as a shallow method, ODIN requires re-initialization for newly introduced nodes and graphs. Mag-Net and SigMaNet also perform robustly by leveraging the magnetic Laplacian, extensively harnessing whole-graph information for embedding optimization. It should be noted, however, that they incur a greater computational complexity due to their reliance on complex-valued matrix operations, a challenge that DUPLEX circumvents with its separation of the amplitude and phase embeddings, as detailed in Appendix D.

We further plot the AUC in subtask 1 against the in/out-degree of nodes in the testing sets of Citeseer and Cora in Figure 3. The x-axis in the figure denotes the threshold, indicating that in the testing set, the source node's out-degree or the target node's in-degree does not exceed the specified threshold. This analysis provides insights into the capability of the benchmark methods in handling low-degree nodes. As expected, the single embedding methods, namely DU-PLEX, MagNet and SigMaNet, are robust to the change of the node degree and consistently outperform the dual embedding methods, namely NERD, DGGAN, DiGAE and ODIN. This superiority is particularly evident when the maximum in/out-degree of testing nodes becomes small (e.g., 1). In this scenario, DUPLEX's performance surges, attributable to its complex conjugate role-based collaborative optimization, yielding increases from a minimum of 25.9% up to 55.0% on Citeseer, and from 16.3% to 47.8% on Cora, when juxtaposed with dual embedding methods. Moreover, DU-PLEX performs even better than MagNet and SigMaNet. They adopt the magnetic Laplacian as the filter to extract features from the digraph and further learn a linear classifier for link prediction based on these features. Instead, DU-PLEX embraces a reconstruction-centric approach, striving to reconstitute the HAM through a direction-aware loss, subsequently guiding the dual encoder towards a deeper assimi-

lation of the graph structure. This reconstruction-oriented methodology likely underpins DUPLEX's enhanced performance relative to feature extraction and linear classification-based methods. In summary, DUPLEX overcomes the problem of suboptimal embedding for low-degree nodes in digraphs, surpassing other methods by large margins.

4.2.2. TRANSDUCTIVE NODE CLASSIFICATION

In this study, we assess the adaptability of node embeddings generated by DUPLEX, which were trained using the proposed self-supervised approach, to secondary tasks such as node classification. After initially training embeddings to encapsulate the graph's structure over the entire graph as delineated in Section 3, we leverage these embeddings to train a two-layer Multilayer Perceptron (MLP) for node classification. It is important to highlight that this experiment unfolds in a transductive setting where labels for the test set are concealed during training, but not the nodes.

Our comparative analysis encompasses **DUPLEX** and a cohort of self-supervised counterparts, including **NERD**, **DGGAN**, **DiGAE**, and **ODIN**. For DUPLEX, we combine amplitude and phase embeddings to derive the final node representations, whereas for the remaining methods, source and target embeddings are concatenated. We also introduce **DUPLEX-S**, a supervised iteration of DUPLEX, expressly trained end-to-end for link prediction, to determine if embeddings optimized for one task maintain their relevance across others. To condition DUPLEX-S on the supervised link prediction task, we adopt the methodology from (Zhang et al., 2021), which involves concatenating complex embeddings for edge-associated node pairs and employing a linear layer as both decoder and classifier. We then concurrently train the dual encoder and decoder to classify edges into four distinct categories. Nodes across all datasets are partitioned into training, validation, and testing subsets with a 3:1:1 split. We utilize macro F_1 and micro F_1-scores for our evaluation metrics.

As demonstrated in Table 5, DUPLEX outperforms other methods on the Citeseer dataset, achieving the highest macro F_1 and micro F_1-scores and exceeding baselines by at least 4.0% and 3.7%, respectively. On Cora-ml, DUPLEX fol-

lows closely behind NERD, securing the second-best results. This indicates that DUPLEX's self-supervised node embeddings are well-suited for node classification tasks. In comparison, the supervised version, DUPLEX-S, lags behind, underscoring the value of the parameter-free decoder and self-supervised learning in improving task generalization. NERD's notable performance can be attributed to its accounting for various node similarity types (both low and high-order). Capturing higher-order proximities indeed helps preserve global network structures more effectively, as nodes with shared neighbors are likely to exhibit similarity. For example, in citation networks, a higher overlap in cited references between two papers often indicates a stronger thematic connection between them. This is particularly advantageous for classification tasks, as highlighted in previous work (Tang et al., 2015). DUPLEX, by stacking GAT layers, captures higher-order proximities effectively, achieving comparable or even superior results to NERD. Additionally, as discussed in detail in Appendix H, NERD employs a more flexible approach to aggregating neighbor information, thus avoiding information bottlenecks and enabling better representation of low-degree nodes, thereby achieving improved node classification performance.

4.2.3. INDUCTIVE NODE CLASSIFICATION

We extend our assessment of DUPLEX to an inductive setting, where testing nodes are entirely unseen during the training phase, a notable departure from our earlier transductive experiment. In this experiment, we conducted a comparative analysis of DUPLEX against three groups of baseline methods. Initially, due to the absence of open-source versions of inductive directed graph embedding methods like BLADE (Virinchi & Saladi, 2023), we contrast **DUPLEX** with transductive models such as **MagNet** and **SigMaNet**. For these models, we retain a transductive framework where only node labels were hidden in the testing set, providing these methods with additional training information. Secondly, we compare DUPLEX with SOTA inductive GNNs designed for directed graph node classification, such as **Dir-GNN** (Rossi et al., 2024). As Dir-GNN can be integrated with different backbone models, we consider SAGE (Hamilton et al., 2017) (denoted as **dir-SAGE**) and GAT (Velickovic et al., 2018) (denoted as **dir-GAT**) here. Thirdly, we pit DUPLEX against SOTA inductive models for undirected graphs, such as the initial undirected **GAT** and **SAGE**, using these comparisons as ablation studies to emphasize the importance of directional information in DUPLEX. GAT and SAGE act as simplified incarnations of DUPLEX, with GAT missing the phase encoder and SAGE serving as the undirected graph aggregator analogue. Our experiments are conducted on Citeseer and Cora-ml, both consisting of attributed graphs. As initial embeddings, we employ the node attributes, which correspond to the word embeddings

Table 6. Inductive node classification results.

Method	Citeseer		Cora-ml	
	mac. F_1	mic. F_1	mac. F_1	mic. F_1
SAGE	70.4(1.2)	74.3(1.0)	74.5(3.2)	81.0(2.4)
GAT	70.8(2.0)	74.7(1.4)	77.1(2.8)	83.9(1.7)
dir-SAGE	70.2(2.6)	74.1(1.2)	83.4(2.2)	85.6(1.9)
dir-GAT	70.4(1.6)	74.6(0.6)	85.0(0.9)	86.4(0.7)
MagNet	65.0(1.7)	74.6(1.9)	82.0(1.7)	84.3(1.5)
SigMaNet	63.1(1.3)	68.8(1.0)	82.7(1.0)	83.5(1.2)
DUPLEX*	68.3(1.3)	74.0(0.7)	85.0(2.6)	87.3(2.5)
DUPLEX	**71.7(0.7)**	**75.4(0.5)**	**85.9(0.8)**	**87.6(0.9)**

of the paper descriptions in these datasets.

Table 6 details the achieved macro and micro F_1-scores, demonstrating DUPLEX's superior performance. Outperforming transductive methods, which benefit from a more informative training phase, DUPLEX reports increases of at least 6.7% and 3.2% in terms of macro F_1-score. In comparison to inductive GNNs, DUPLEX shows enhancements of at least 1.3% and 0.9% for directed graph methods and at least 0.9% and 8.8% for undirected graph methods. These results underscore DUPLEX's robust inductive capabilities, adeptly managing unseen nodes and accurately classifying instances across a spectrum of categories. Furthermore, DUPLEX's significant outperformance over baselines on undirected graphs emphasizes its efficiency in utilizing directional information, an attribute integral to tasks involving directed graphs.

4.3. Ablation Study

The results of our ablation study are summarized below, with comprehensive details provided in Appendix J: (1) **Dual GAT Encoder**: The amplitude and phase aggregators are essential for capturing both the connections and the directions within the graph. The dual encoder confers a performance advantage irrespective of the backbone model. (2) **Fusion Layer**: The "mid-fusion" approach is preferred for graphs lacking attributes; "Fusion aggregation" in Eq. (9) demonstrates superioriority over a strategy that involves two distinct steps. (3) **Sensitivity Analysis**: The inclusion of the connection-aware loss can enhance the performance, but the enhancement decays when the weight λ increases.

5. Conclusion

We propose DUPLEX for digraph embedding that yields high-quality embeddings especially for low in/out-degree nodes, adapts well to diverse downstream tasks, and generalizes effectively to unseen nodes. DUPLEX achieves these objectives through a dual encoder, two parameter-free decoders, and two self-supervised losses. Results show that it surpasses SOTA methods in link prediction, as well as transductive and inductive node classification tasks.

Acknowledgements

Impact Statement

This paper presents work whose goal is to advance the field of graph neural networks. There are many potential societal consequences of our work, though we believe it is unnecessary to highlight them in this context.

References

Fiorini, S., Coniglio, S., Ciavotta, M., and Messina, E. Sigmanet: one laplacian to rule them all. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Guo, K. and Mohar, B. Hermitian adjacency matrix of digraphs and mixed graphs. *Journal of Graph Theory*, 85, 2015.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Hohn, F. E. *Elementary matrix algebra*. Courier Corporation, 2013.

Hu, P., Chan, K. C., and He, T. Deep graph clustering in social network. In *Proceedings of the 26th international conference on world wide web companion*, pp. 1425–1426, 2017.

Khosla, M., Leonhardt, J., Nejdl, W., and Anand, A. Node representation learning for directed graphs. In *Machine Learning and Knowledge Discovery in Databases*, pp. 395–411, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Ko, T., Choi, Y. K., and Kim, C.-K. Universal graph contrastive learning with a novel laplacian perturbation. In *Uncertainty in Artificial Intelligence*, 2023.

Kollias, G., Kalantzis, V., Id'e, T., Lozano, A. C., and Abe, N. Directed graph auto-encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Lin, L. and Gao, J. A magnetic framelet-based convolutional neural network for directed graphs. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1–5, 2023.

Liu, J. and Li, X. Hermitian-adjacency matrices and hermitian energies of mixed graphs. *Linear Algebra and its Applications*, 466:182–207, 2015.

Liu, L., Chen, K.-J., and Liu, Z. Collaborative bi-aggregation for directed graph embedding. *Neural Networks*, 164:707–718, 2023.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Robert, C. P., Casella, G., and Casella, G. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

Rossi, E., Charpentier, B., Di Giovanni, F., Frasca, F., Günnemann, S., and Bronstein, M. M. Edge directionality improves learning on heterophilic graphs. In *Learning on Graphs Conference*, pp. 25–1, 2024.

Salha-Galvan, G., Limnios, S., Hennequin, R., Tran, V.-A., and Vazirgiannis, M. Gravity-inspired graph autoencoders for directed link prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, 2015.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Virinchi, S. and Saladi, A. Blade: biased neighborhood sampling based graph neural network for directed graphs. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023.

Šubelj, L. and Bajec, M. Model of complex networks based on citation dynamics. In *Proceedings of the 22nd International Conference on World Wide Web*, pp. 527–530, 2013.

Wu, A. and Han, Y. Multi-modal circulant fusion for video-to-language and backward. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 1029–1035, 2018.

Wu, S., Sun, F., Zhang, W., Xie, X., and Cui, B. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Yang, S., Zhang, Z., Zhou, J., Wang, Y., Sun, W., Zhong, X., Fang, Y., Yu, Q., and Qi, Y. Financial risk analysis for smes with graph-based supply chain mining. In

Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence, pp. 4661–4667, 2021.

Yoo, H., Lee, Y.-C., Shin, K., and Kim, S.-W. Disentangling degree-related biases and interest for out-of-distribution generalized directed network embedding. In *Proceedings of the 32nd International Conference on World Wide Web*, pp. 231–239, 2023.

Yu, H., Wu, S., and Dauwels, J. Efficient variational bayes learning of graphical models with smooth structural changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):475–488, 2022.

Yu, Z., Yu, J., Fan, J., and Tao, D. Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *2017 IEEE International Conference on Computer Vision*, pp. 1839–1848, 2017.

Yue, X., Wang, Z., Huang, J., Parthasarathy, S., Moosavinasab, S., Huang, Y., Lin, S. M., Zhang, W., Zhang, P., and Sun, H. Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics*, 36(4):1241–1251, 2020.

Zhang, X., He, Y., Brugnone, N., Perlmutter, M., and Hirn, M. J. Magnet: a neural network for directed graphs. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, volume 34, pp. 27003–27015, 2021.

Zheng, C., Fan, X., Wang, C., and Qi, J. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Zhou, C., Liu, Y., Liu, X., Liu, Z., and Gao, J. Scalable graph embedding for asymmetric proximity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Zhou, H., Chegu, A., Sohn, S. S., Fu, Z., de Melo, G., and Kapadia, M. D-hypr: harnessing neighborhood modeling and asymmetry preservation for digraph representation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2021.

Zhu, S., Li, J., Peng, H., Wang, S., Yu, P. S., and He, L. Adversarial directed graph embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

A. Symbols and Notations

Table 7. Symbols and notations used in this paper.

Symbols	Notations
$\mathcal{G}$	a digraph with nodes and edges
$\mathcal{V}$	the set of nodes in the graph
$\mathcal{E}$	the set of edges in the graph
u, v	two nodes on the graph for illustration
$\boldsymbol{A}$	the adjacency matrix of graph
$\boldsymbol{x}_u$	the embedding vector of nodes u
$\boldsymbol{X}$	the embedding matrix of nodes
$\boldsymbol{H}$	the Hermitian adjacency matrix of graph
$\boldsymbol{A}_s$	the amplitude matrix of HAM
$\boldsymbol{\Theta}$	the phase matrix of HAM
$\boldsymbol{a}_u$	the amplitude embedding of nodes u
$\boldsymbol{\theta}_u$	the phase embedding of nodes u
$\mathcal{N}(u)$	the neighbors of node u
$\sigma(\cdot)$	the sigmoid function
$\phi(\cdot)$	the activation function
$\mathcal{R}$	four relation types of edge
r	a certain relation type of edge
q	the decay rate of the weight of connection-aware loss
λ	the initial weight of the connection-aware loss
d	embedding dims

B. Limitations of Dual Embedding Methods: An SVD Perspective

We address a specific challenge that dual embedding methods face with nodes exhibiting low in-/out-degrees and rationalize this from a Singular Value Decomposition (SVD) standpoint. In this section, we introduce Lemma B.1 (see (Hohn, 2013)) and Lemma B.2, provide a proof of Lemma B.2, and summarize our insights.

Lemma B.1 (Sylvester's rank inequality). *For matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times s}$ with $\boldsymbol{AB} = \boldsymbol{0}$, it holds that $r(\boldsymbol{A}) + r(\boldsymbol{B}) \leq n$.*

In adjacency matrices, nodes with zero out-degree correspond to rows filled with zeros, while zero in-degree nodes match with columns of zeros. Focusing on the zero out-degree nodes, we establish that:

Lemma B.2. *If an asymmetric matrix has several all-zero rows, the corresponding rows in its source embedding matrix are all-zero.*

Proof. Let $\boldsymbol{A}$ be an asymmetric matrix with several zero rows. We can represent it as follows by renumbering nodes, that is, by interchanging the rows and columns:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}' \\ \boldsymbol{0} \end{bmatrix}, \tag{13}$$

where $\boldsymbol{A}'$ is a submatrix of $\boldsymbol{A}$ consisting of all non-zero rows. We further perform the singular value decomposition (SVD) on $\boldsymbol{A}$:

$$\begin{bmatrix} \boldsymbol{A}' \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_2 \end{bmatrix} \Sigma \boldsymbol{V}^T, \tag{14}$$

where

$$\boldsymbol{A}' = \boldsymbol{U}_1 \Sigma \boldsymbol{V}^T, \tag{15}$$

12

and

$$\mathbf{0} = \boldsymbol{U}_2 \Sigma \boldsymbol{V}^T. \tag{16}$$

It is important to note that $\boldsymbol{A}'$ is the predominantly optimized component, encompassing the majority of the training samples. Specifically, our emphasis lies on optimizing Eq. (15) by adjusting its singular values and vectors. In the context of low-rank decomposition, we consider the first d largest singular values, which are both non-zero and distinct, thus the corresponding singular vectors are orthogonal.

With Lemma B.1, we have:

$$r(\boldsymbol{U}_2) + r(\Sigma \boldsymbol{V}) <= d, \tag{17}$$

where $r(\Sigma \boldsymbol{V}) = d$ as the column vectors of $\boldsymbol{V}$ are orthogonal. Consequently, we deduce that $r(\boldsymbol{U}_2) = \mathbf{0}$, that is,

$$\boldsymbol{U}_2 = \mathbf{0}. \tag{18}$$

For the dual embedding methods, the source embedding matrix can be written as

$$\boldsymbol{X}_s = \begin{bmatrix} \boldsymbol{U}_1 \\ \boldsymbol{U}_2 \end{bmatrix} \Sigma^{1/2} = \begin{bmatrix} \boldsymbol{U}_1 \\ \mathbf{0} \end{bmatrix} \Sigma^{1/2}, \tag{19}$$

that is, the corresponding rows in the source embedding matrix are all-zero. $\qquad\square$

Similarly, it can be shown that an asymmetric matrix with columns of zeros will have target embedding matrix columns filled with zeros, leading to zero target/source embeddings for nodes with zero in-/out-degrees. Examining nodes with low in-/out-degrees, and taking low out-degree nodes as an example, the adjacency matrix is divided into high and low out-degree segments:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}' \\ \boldsymbol{O} \end{bmatrix}, \tag{20}$$

with $\boldsymbol{A}'$ representing high out-degree nodes and $\boldsymbol{O}$ for low out-degree nodes. Suppose that errors are uniformly distributed across both segments during optimization (i.e., reconstruction of $\boldsymbol{A}$), and these errors set non-zero entries in $\boldsymbol{A}$ to zero. This affects low-degree nodes more significantly, potentially pushing their embeddings toward zero and causing them to cluster in the embedding space, as cautioned by Lemma B.2. Although this issue is not rigorously proven for low-degree nodes and remains a topic for further exploration, we can assert that the HAM does not suffer from this problem as it has all-zero rows or columns only for isolated nodes with both zero in and out-degree, which are outside the scope of our analysis.

C. Graph Attention Networks

GAT (Velickovic et al., 2018) is a successful practice of incorporating attention mechanisms into graph neural networks. Specifically, GAT assigns attention coefficients α_{uv} to each connected pair of nodes (u, v) as weights during neighbor aggregation. In DUPLEX, we use GAT as our backbone model for both amplitude and phase encoders. Taking the amplitude GAT as an example, we further write Eq. (6) as:

$$\sum_{v \in \mathcal{N}(u)} f_a(\boldsymbol{a}_u, \boldsymbol{a}_v)\psi_a(\boldsymbol{a}_v) = \sum_{v \in \mathcal{N}(u)} \alpha_{uv} \boldsymbol{W}_a \boldsymbol{a}_v, \tag{21}$$

where the attention coefficient,

$$\alpha_{uv} = \frac{\exp(\text{LeakyReLU}(\boldsymbol{b}_a^\intercal [\boldsymbol{W}_a \boldsymbol{a}_u || \boldsymbol{W}_a \boldsymbol{a}_v]))}{\sum_{g \in \mathcal{N}(u)} \exp(\text{LeakyReLU}(\boldsymbol{b}_a^\intercal [\boldsymbol{W}_a \boldsymbol{a}_u || \boldsymbol{W}_a \boldsymbol{a}_g]))}. \tag{22}$$

Here, $\exp(\cdot)$ is the exponential operation, $\text{LeakyReLU}(\cdot)$ is the activation function, $||$ is the concatenation operation, $\boldsymbol{b}_a \in \mathbb{R}^{2d \times 1}$ is the attention parameters of the amplitude encoder, and $\boldsymbol{W}_a \in \mathbb{R}^{d \times d}$ is the feature transformation parameters.

D. Complexity Analysis

Assuming the network has L layers in total, the embedding dimension is d, with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges in the graph. In our method, for each head of the attention mechanism, the time complexity of the encoding process is $O(L|\mathcal{V}|d^2 + L|\mathcal{E}|d)$, and the space complexity is $O(|\mathcal{E}| + Ld^2 + L|\mathcal{V}|d + Ld)$. The time complexity of the decoding process is $O(|\mathcal{E}|d)$, and the space complexity is $O(|\mathcal{V}|d + |\mathcal{E}|)$. The total number of parameters in the model is $2Ld^2 + 4Ld$.

We further compare the dual GAT encoder of DUPLEX with a single GAT encoder that directly produces a d-dimensional vector of complex numbers for each node in each layer, where the complex vector is often converted to a $2d$-dimensional real-valued vector without separating the amplitude and phase. The complexity of feature transformation in each layer of the dual GAT encoder, encompassing both amplitude and phase transformations, amounts to $2|\mathcal{V}|d^2$. In contrast, for the single GAT encoder, the complexity for the $|\mathcal{V}| \times 2d$ feature matrix is $4|\mathcal{V}|d^2$. This reveals that our dual GAT encoder reduces feature transformation complexity by half, while maintaining the same complexity for other operations.

E. More Details on Datasets

We perform our experiments on five open-source digraph datasets, including Cora-ml, Citeseer, Cora, Epinions and Twitter. **Citeseer** and **Cora** are two popular citation networks, **Cora-ml** is a subset of Cora dataset in the field of machine learning. The Cora-ml and Citeseer datasets provide meaningful node features, with the node labels corresponding to scientific subareas. **Epinions** and **Twitter** are two social networks. We use the version of **Citeseer** and **Cora-ml** dataset provided by (Zhang et al., 2021), **Cora** dataset provided by (Šubelj & Bajec, 2013) and other two datasets from the Stanford Large Network Dataset Collection[2]. The overview of the datasets is shown in Table 2. The graph datasets used in our study are unweighted and directed. These datasets encompass a wide range of graph sizes, with the number of nodes varying from $3K$ to $465K$, and the number of edges ranging from $4K$ to $834K$.

The direction information is quite important in these datasets. For example, consider two nodes in a citation network. Node a has many in-neighbors, indicating that it is a popular paper cited by many other papers. On the other hand, node b has many out-neighbors, indicating that it is a knowledgeable paper that cites many other papers. These two papers have completely different characteristics. However, when using undirected graph embedding methods, as both nodes are connected to many other nodes, they would be considered to have similar structural features in the graph, which can mislead downstream tasks. Similarly, in a social network, if A follows several people who all follow B, it is more likely that A will follow B rather than the other way around. Undirected graph embedding methods would consider A and B to have similar structural features and fail to distinguish their asymmetrical relationship, leading to potential errors in recommendations and other tasks.

F. Implementation Details

We compare DUPLEX with several SOTA models, where NERD, DGGAN, DiGAE and ODIN learn a dual embedding while DUPLEX, MagNet and SigMaNet learn a single embedding for each node. During experiments, the dual embedding methods learn two 128-dimensional real-valued embeddings for each node, while the single embedding methods learn a 128-dimensional complex embedding for each node.

We implemented DUPLEX using DGL and PyTorch, employing Adam optimizer with a learning rate of 1e-3. DUPLEX consists of two 3-layer GATs with attention head 1, one for the amplitude embedding and the other for the phase embedding. We sampled four types of node pairs (forward edge, reverse edge, bidirectional edge, and no edge) at a ratio 1:1:1:x for self-supervised loss computation, where x tends to be small due to fewer bidirectional edges, and in our experiments we just control $x \leq 1$. We set the hidden dim to 128 and the dropout rate to 0.5. We tuned the initial loss weight λ in $\{0.1, 0.3\}$ and the decay rate q in $\{0, 1e\text{-}4, 1e\text{-}2\}$. We ran our model with maximum 3000 epochs with early stopping for all experiments.

For the baseline methods, we utilize the publicly available code repositories, and tune the hyperparameters as follows.

- NERD is a random-walk based model, for which we tune the walk size in $\{1, 4, 7\}$, the start learning rate in $\{0.001, 0.01, 0.1\}$ and the negative samples in $\{1, 4, 7\}$.

- DGGAN is a GAN-based model, for which we set the default parameters as the paper.

- ODIN is a shallow method incorporating multiple losses with distinct objectives, where we tune the disentangling loss

---

[2]https://snap.stanford.edu/data/

weights in $\{0.3, 0.5, 0.7\}$ and the negative sampling rates in $\{1, 4, 7\}$.

- MagNet is a GCN-based model, where we set the GCN layer as 3, hidden dim as 128, and tune the parameter q, which controls the importance of directed information, in $\{0.05, 0.1, 0.15, 0.2, 0.25\}$.

- DiGAE is a digraph autoencoder model which set directed GCN as the encoder, for which we tune the hyperparameter α, β in $\{0.3, 0.5, 0.7\}$.

- SigMaNet is a GCN-based model, where we set the GCN layer as 3 and the hidden dim as 128.

G. Link Prediction Set-up

In Section 4.2.1, we assess the model's ability to discriminate between different edge types and consider four commonly-used subtasks. We list the details of each subtask here.

1. Existence Prediction (EP) (Zhu et al., 2020): This subtask involves predicting whether a specific edge exists in the graph or not. In the testing sets, the ratio of edges belonging to the two types is 1:1. Note that both reverse and non-existent edges are categorized as non-existence (Zhu et al., 2020), and their ratio is also 1:1 in the testing set.

2. Direction Prediction (DP) (Zhu et al., 2020; Zhang et al., 2021): The objective of this subtask is to predict the direction of unidirectional edges (i.e., forward or reverse), with a ratio of 1:1 in the testing sets.

3. Three-type Classification (TP) (Zhang et al., 2021): This subtask involves classifying edges into three types: positive, reverse, or non-existent, with a ratio of 1:1:1 in the testing set.

4. Four-type Classification (FP): In this subtask, the goal is to classify edges into four categories: positive, reverse, bidirectional, or non-existent. The ratio for the four types of edges in the testing set is set to 1:1:1:1. In cases where the number of bidirectional edges is smaller than the other types in certain datasets, we utilize all available bidirectional edges in the testing set.

H. Node Classification Performance

The ease of classifying nodes is closely tied to their degrees. It is widely acknowledged that nodes with ample accessible information, whether emanating from structural linkages or intrinsic nodal attributes, are typically more amenable to effective classification. When nodes are bolstered by reliable attributes, these specific individualistic features can significantly diminish the relative importance of structural information in achieving separability between different node classes. Consequently, even nodes with a low degree have the potential to be accurately classified based solely on their distinctive attributes. In contrast, when such nodal attributes are absent or lack distinctive information, the graph's structural properties gain prominence as a critical discriminative element. In this context, nodes characterized by a scant number of connections, or a low in/out degree, inherently face challenging classification scenarios attributed to their sparse connectivity within the overall graph structure.

Addressing these challenges, DUPLEX brings forth a flexible encoding approach that dynamically assimilates information within the graph. Recall that one common embedding method for directed graphs employs the source and target embeddings (Khosla et al., 2019; Kollias et al., 2022; Yoo et al., 2023). To finish the node classification task, the concatenation of the source and target embeddings is used, that is, $[s, t]$. We notice that typically $s = f(x_{\text{out}})$ and $t = f(x_{\text{in}})$. In other words, the source embeddings only aggregate information from the out-neighbors, while the target embeddings only aggregate information from the in-neighbors. Information from both types of neighbors can only be exchanged in the final concatenation stage. By contrast, DUPLEX's design innovatively facilitates information fusion between both types of neighbors at every layer. This occurs through the coordinated use of its amplitude encoder with an undirected graph aggregator and its phase encoder with a directed graph aggregator, circumventing the information bottleneck commonly found in previous approaches.

Furthermore, since we employ the attention mechanism for the sake of aggregation, it can learn the attention score between two nodes adaptively from the data. When this score approaches zero, the information exchange is blocked. For graphs with reliable attributes, DUPLEX possesses the capability to truncate the exchange of information between nodes by modulating attention scores that approach zero and sorely relies on the node attributes to achieve good classification performance. Conversely, in the case when the graph structure plays a more important role, DUPLEX can aggregate information from both in and out neighbors by maintaining higher attention scores, aiding low-degree nodes in overcoming their inherent informational disadvantage.

I. The Convergence of Self-supervised Training

The training of DUPLEX via self-supervised loss relies on the principles of stochastic gradient descent (SGD), a well-established optimization method with established theoretical foundations ensuring convergence under mild conditions. Specifically, based on the theory of stochastic gradient descent (Robert et al., 1999), using a schedule of the learning rates ρ_t such that $\sum \rho_t = \infty$ and $\sum \rho_t^2 < \infty$, the training process will converge to a local minimum of the DUPLEX loss function.

In addition to this theoretical assurance, we capture and illustrate the progression of the loss function's convergence over the course of the training period through visual plots. Likewise, we track the mean square error between the HAM approximated by DUPLEX and the actual ground truth HAM, with these findings detailed in Figure 4. The graphical representations of these metrics clearly depict the effective convergence of the model during training, which corroborates the theoretical underpinnings of our approach.
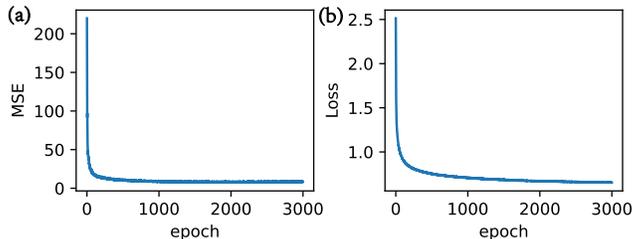
Figure 4. (a) Loss curve. (b) Mean square error between approximated HAM and the ground truth.

J. Ablation Study

In this section, we perform ablation studies to delve into analyzing the role of each component in DUPLEX.

J.1. Dual GAT Encoder

We consider two variants of the DUPLEX: **DUPLEX-am**, which employs the undirected graph aggregator $\oplus$ (see Eq. (6)) for both amplitude and phase embeddings, and **DUPLEX-ph**, which instead utilizes the digraph aggregator $\vec{\oplus}$ (see Eq. (7)) for both types of embeddings. We conduct the four subtasks of link prediction on the Citeseer and Cora datasets, following the same settings in Section 4.2.1. As shown in Table 8, DUPLEX outperforms both DUPLEX-am and DUPLEX-ph, underscoring the necessity of using the directed and undirected graph aggregator respectively for phase and amplitude embeddings. DUPLEX-am performs poorly in all tasks due to the omission of direction. Interestingly, DUPLEX-ph performs well in the direction prediction subtask, suggesting the effectiveness of the proposed directed aggregator for capturing the direction information.

Table 8. Link prediction ACC and AUC (%) of four subtasks. The best results are in **bold** and the second are <u>underlined</u>.

Dataset	Method	EP		DP		TP	FP
		AUC	ACC	AUC	ACC	ACC	ACC
Citeseer	DUPLEX-am	86.2(3.9)	79.3(5.1)	93.2(1.1)	88.4(1.9)	73.4(2.7)	67.7(3.0)
	DUPLEX-ph	<u>93.7(1.2)</u>	<u>91.8(1.2)</u>	**99.8(0.0)**	**99.0(0.1)**	<u>82.9(1.0)</u>	<u>80.8(1.0)</u>
	DUPLEX	**98.6(0.4)**	**95.7(0.5)**	<u>99.7(0.2)</u>	<u>98.7(0.4)</u>	**94.8(0.2)**	**91.1(1.0)**
Cora	DUPLEX-am	89.5(0.2)	87.0(0.2)	88.8(0.8)	86.7(0.3)	85.1(1.3)	81.8(1.8)
	DUPLEX-ph	<u>95.2(0.1)</u>	<u>91.2(0.1)</u>	**98.4(0.2)**	**97.2(0.2)**	<u>88.3(0.2)</u>	<u>83.7(0.7)</u>
	DUPLEX	**95.9(0.1)**	**93.2(0.1)**	<u>97.2(0.2)</u>	<u>95.9(0.1)</u>	**92.2(0.1)**	**88.4(0.4)**

Moreover, to demonstrate the robustness of the dual encoder design with respect to different backbone GNNs, we replace the GAT backbone with the spatial GCN (Kipf & Welling, 2017; Hamilton et al., 2017), where the aggregation function is:

$$\oplus\big(\{\boldsymbol{a}_v, \forall v \in \mathcal{N}(u)\}\big) = \sum_{v \in \mathcal{N}(u)} \psi_a(\boldsymbol{a}_v), \tag{23}$$

16

and

$$\vec{\oplus} \left(\{\boldsymbol{\theta}_v, \forall v \in \mathcal{N}(u)\} \right) = \sum_{v \in \mathcal{N}_{\text{in}}(u)} \psi_\theta(\boldsymbol{\theta}_v) - \sum_{v \in \mathcal{N}_{\text{out}}(u)} \psi_\theta(\boldsymbol{\theta}_v). \tag{24}$$

The difference is that there is no need for computation of attention coefficients.

We repeat our experiments concerning link prediction, transductive and inductive node classification. The experiment results are shown in Table 9-11. It can be observed that DUPLEX with a GAT backbone consistently outperforms the counterpart with a GCN backbone, due to the higher flexibility of GAT. However, even with the GCN backbone, DUPLEX still surpasses other SOTA methods across various tasks, highlighting the advantage of the dual encoder design regardless of the backbone.

Table 9. Link prediction ACC (%) for four subtasks. The best results are in **bold** and the second are underlined. Note that 'EP' represents the 'Existence Prediction' subtask, while 'DP' for direction Prediction, 'TP' for three-type classification and 'FP' for four-type classification.

Method	Cora				Epinions			
	EP	DP	TP	FP	EP	DP	TP	FP
DUPLEX*(GCN)	92.2(0.3)	95.2(0.2)	90.8(0.2)	86.8(0.0)	83.3(0.2)	91.8(0.1)	87.3(0.5)	73.1(0.1)
DUPLEX*(GAT)	93.1(0.1)	95.5(0.3)	92.1(0.1)	88.3(0.2)	84.5(0.5)	92.2(0.0)	88.9(0.1)	74.9(0.6)
DUPLEX(GCN)	93.0(0.1)	95.6(0.1)	91.9(0.3)	88.1(0.4)	84.7(0.2)	92.5(0.0)	87.6(0.3)	75.2(0.2)
DUPLEX(GAT)	**93.2(0.1)**	**95.9(0.1)**	**92.2(0.1)**	**88.4(0.4)**	**85.5(0.0)**	**92.6(0.1)**	**88.9(0.0)**	**76.4(0.2)**

Table 10. Transductive Node classification result (%) with self-supervised training.

Method	Citeseer		Cora-ml	
	mac. F_1	mic. F_1	mac. F_1	mic. F_1
DUPLEX*(GCN)	49.0(1.3)	52.1(1.1)	73.9(0.3)	76.2(0.3)
DUPLEX*(GAT)	51.2(0.1)	54.3(2.0)	76.0(1.3)	77.8(1.6)
DUPLEX(GCN)	51.2(1.0)	54.0(1.0)	75.5(0.7)	77.7(0.5)
DUPLEX(GAT)	**53.0(2.5)**	**56.2(1.7)**	**77.9(0.6)**	**79.8(0.7)**

Table 11. Inductive node classification macro F_1 (%) and micro F_1 (%) with supervised training.

Method	Citeseer		Cora-ml	
	mac. F_1	mic. F_1	mac. F_1	mic. F_1
DUPLEX*(GCN)	70.9(0.6)	74.9(0.9)	82.2(1.5)	84.8(1.1)
DUPLEX*(GAT)	68.3(1.3)	74.0(0.7)	85.0(2.6)	87.3(2.5)
DUPLEX(GCN)	68.9(2.7)	74.7(0.9)	82.6(2.1)	85.0(1.1)
DUPLEX(GAT)	**71.7(0.7)**	**75.4(0.5)**	**85.9(0.8)**	**87.6(0.9)**

J.2. Fusion Layer

We can tell that the fusion layer can enhance the performance of DUPLEX across all tasks and datasets by comparing DUPLEX with DUPLEX* in Tables 3-6, implying the importance of utilizing the complementary information in the amplitude and phase encoder. Despite this, DUPLEX* still achieves commendable results, competing favorably against benchmark methods. This empirical outcome underscores the robustness of DUPLEX's design, even when the encoders operate in isolation without information fusion.

We further investigate the impact of the placement of the fusion layer on the performance of DUPLEX, considering four strategies: **early-fusion**, **mid-fusion**, **late-fusion**, and **all-fusion**. These strategies involve applying the fusion operation defined in Eq. (9) at the input, middle, output, and all layers, respectively. Our investigation includes both non-attributed and attributed graphs, focusing on the task of node classification. Notably, in non-attributed graphs, the node embeddings are randomly initialized and trained in a self-supervised manner, while in attributed graphs, they are initialized using the node attributes and trained in a fully supervised manner. The results are shown in Figs. 5-6. For non-attributed graphs, the mid-fusion strategy achieves the best performance, in agreement with the analysis discussed in Section 3.2.3. On the other hand, for attributed graphs, the all-fusion strategy yields the highest performance. In this scenario, early-fusion does not introduce noise, as the node attributes hold meaningful information. Similarly, late-fusion does not compromise performance, given that the model is fully supervised for node classification, and the independence of amplitude and phase embeddings is not necessary. Consequently, fusion at all layers effectively integrates the valuable node attributes with the graph structure, resulting in improved performance.

Additionally, to see whether integrating fusion with aggregation behaves better than separating fusion and aggregation operations, we explore another commonly-used fusion approach, known as the element-wise sum (Wu & Han, 2018; Yu et al., 2017). This approach computes a weighted combination of the amplitude embedding and phase embedding of each layer: $a'_u = \phi\big(a_u + \psi_{a\theta}(\theta_u)\big)$. Unlike our fusion method, this approach is independent of the graph structure and fuses
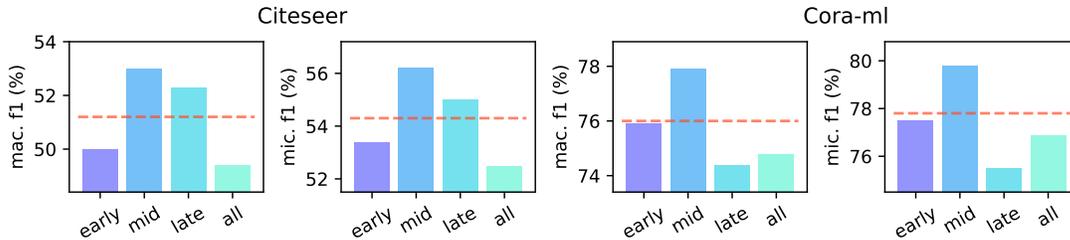
Figure 5. Node classification macro F_1 (%) and micro F_1 (%) with self-supervised training on randomly initialized graphs. The 'early' represent for the 'early-fusion' strategy, 'mid' for 'mid-fusion', 'late' for 'late-fusion', 'all' for 'all-fusion'. The dashed line is the baseline result with no fusion layer.
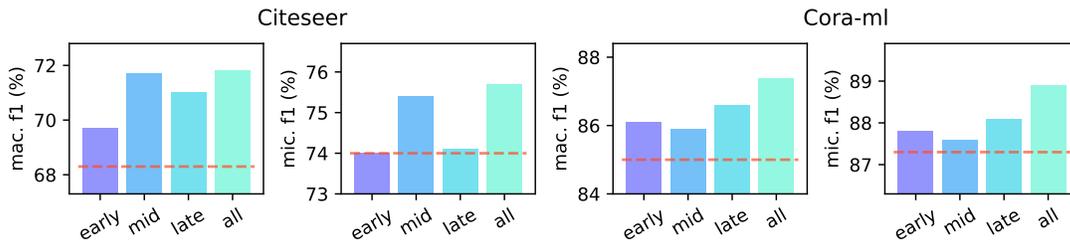
Figure 6. Node classification macro F_1 (%) and micro F_1 (%) with supervised training on feature-initialized graphs. The 'early' represent for the 'early-fusion' strategy, 'mid' for 'mid-fusion', 'late' for 'late-fusion', 'all' for 'all-fusion'. The dashed line is the baseline result with no fusion layer.

the amplitude and phase embeddings of the same node after aggregation in each layer. The results for both transductive and inductive node classification can be found in Table 12-13. We can see that DUPLEX with the proposed fusion method consistently outperforms the element-wise sum approach. This superiority arises from its ability to better exploit the graph structure during fusion, facilitating a more effective exchange of information.

Table 12. Transductive Node classification result (%) with self-supervised training. The DUPLEX(EMS) represents DUPLEX with element-wise sum fusion.

Method	Citeseer		Cora-ml	
	mac. F_1	mic. F_1	mac. F_1	mic. F_1
DUPLEX*	51.2(0.1)	54.3(2.0)	76.0(1.3)	77.8(1.6)
DUPLEX(EWS)	52.7(1.1)	56.1(2.0)	74.6(2.9)	76.7(2.5)
DUPLEX	**53.0(2.5)**	**56.2(1.7)**	**77.9(0.6)**	**79.8(0.7)**

Table 13. Inductive node classification macro F_1 (%) and micro F_1 (%) with supervised training. The DUPLEX(EMS) represents DUPLEX with element-wise sum fusion.

method	Citeseer		Cora-ml	
	mac. F_1	mic. F_1	mac. F_1	mic. F_1
DUPLEX*	68.3(1.3)	74.0(0.7)	85.0(2.6)	87.3(2.5)
DUPLEX(EMS)	57.1(1.0)	62.7(1.0)	66.6(2.4)	67.8(1.8)
DUPLEX	**71.7(0.7)**	**75.4(0.5)**	**85.9(0.8)**	**87.6(0.9)**

J.3. Sensitivity Analysis

To check the sensitivity of the initial loss weight λ and the decay rate q in the loss function (see Section 3.3.3), we conduct experiments by varying the loss weight λ from 0.0 to 1.0 and the decay rate q in the set $\{0, 1e\text{-}4, 1e\text{-}2\}$. The results for link prediction and transductive node classification are presented in Figure 7-8. Our findings indicate that incorporating a non-zero weight $\lambda > 0$ for the connection-aware loss results in improved performance compared to having no connection-aware loss ($\lambda = 0$). However, as the initial weight λ increases, the performance of DUPLEX begins to decrease. Additionally, increasing the decay rate q reduces the influence of different initial weights on the model's performance. These observations suggest that the connection-aware loss aids in the learning of superior embeddings. However, setting the initial loss weight too high can interfere with the primary classification tasks, as the optimization objectives for the two losses are not entirely aligned.
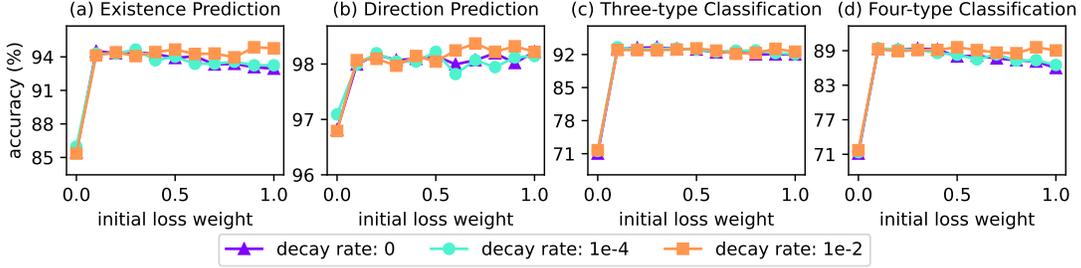
Figure 7. Link prediction accuracy (%) under different initial loss weights and decay rates.
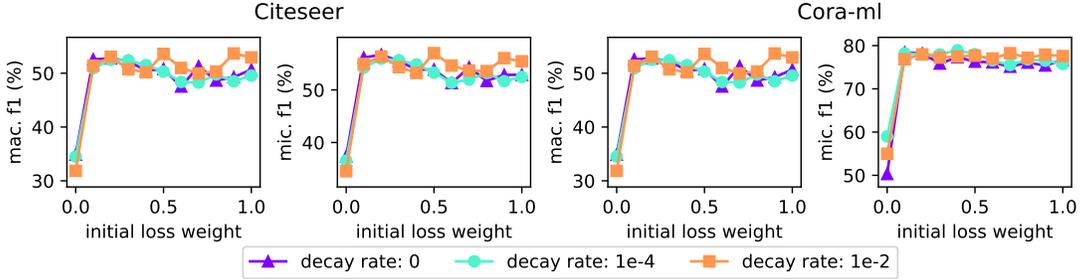
Figure 8. Self-supervised node classification macro F_1 (%) and micro F_1 (%) under different initial loss weights and decay rates.

J.4. Distance Metrics

In Eq. (10), we employ the L1 distance as a measure to quantify the difference between the reconstructed matrix element and the corresponding element in the ground truth Hermitian adjacency matrix. However, it is worth noting that DUPLEX is robust to different distance metrics. In this section, we specifically utilize the L2 distance as an alternative distance metric and conduct link prediction experiments to compare the performance of DUPLEX using L2 distance. Table 14 illustrates the link prediction performance of DUPLEX utilizing L1 distance and L2 distance. From our observations, we find that DUPLEX with L1 distance generally performs slightly better than DUPLEX with L2 distance in most of the experiments conducted. Nevertheless, DUPLEX with L2 distance consistently demonstrates superior performance compared to other methods across all the experiments conducted.

Table 14. Link prediction ACC (%) for four subtasks. Note that 'EP' represents the 'Existence Prediction' subtask, while 'DP' for direction Prediction, 'TP' for three-type classification and 'FP' for four-type classification.

Distance	Citeseer				Cora			
Metric	EP	DP	TP	FP	EP	DP	TP	FP
DUPLEX(L1)	**95.7(0.5)**	**98.7(0.4)**	94.8(0.2)	**91.1(1.0)**	**93.2(0.1)**	**95.9(0.1)**	**92.2(0.1)**	**88.4(0.4)**
DUPLEX(L2)	95.0(0.6)	98.6(0.5)	**95.4(0.6)**	89.5(1.7)	92.6(0.1)	95.2(0.2)	92.1(0.1)	84.6(0.2)
Distance	Epinions				Twitter			
metric	EP	DP	TP	FP	EP	DP	TP	FP
DUPLEX(L1)	85.5(0.0)	**92.6(0.1)**	**88.9(0.0)**	76.4(0.2)	98.7(0.1)	**99.8(0.0)**	**99.2(0.1)**	98.1(0.2)
DUPLEX(L2)	**86.5(0.1)**	92.1(0.0)	88.8(0.1)	**76.7(0.1)**	**98.9(0.1)**	99.6(0.0)	99.2(0.2)	**98.8(0.2)**