



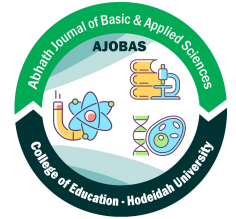
Abhath Journal of Basic and Applied Sciences, 4(2) (2025), 20–39

Abhath Journal of Basic and Applied Sciences

Website: <https://ojs.abhath-ye.com/index.php/OJSRJBAS/about>

doi: <https://doi.org/10.59846/ajobas.v4i2.832>

Research paper



Graph Neural Networks for Bidirected and Multidirected Graphs: A Formal Review

Takaaki Fujita^{1,*}

¹Independent Researcher, Shinjuku, Shinjuku-ku, Tokyo, Japan.

*Corresponding author: E-mail: Takaaki.fujita060@gmail.com

Received: 19 June 2025 Accepted: 27 June 2025 Published: 31 December 2025

Abstract

Graph-structured data often exhibit complex edge semantics beyond simple undirected links—such as one-way flows, independent endpoint orientations, or multiple parallel arcs. While standard Graph Neural Networks (GNNs) handle undirected and directed graphs, they cannot fully exploit richer topologies like bidirected or multidirected graphs. We introduce two novel GNN variants: *Bidirected GNNs*, which separately aggregate messages according to endpoint-specific arrow assignments, and *Multidirected GNNs*, which incorporate parallel-edge counts into feature updates. We present formal definitions, derive update rules, and showcase illustrative examples to demonstrate each model's ability to encode complex orientation and multiplicity patterns. Our frameworks pave the way for more expressive graph representation learning in domains requiring fine-grained edge semantics.

Keywords: Graph Neural Networks (GNNs), Directed Graph Neural Network, Bidirected Graph Neural Network, Multidirected Graph Neural Network, Bidirected Graph, Multidirected graph, Graph Theory

1 Introduction

1.1 Graph Neural Networks

Graph theory offers a flexible framework for modeling pairwise relations by representing entities as vertices and connections as edges [1]. While undirected graphs assume mutual links, more expressive variants capture directionality and multiplicity. Directed graphs assign each edge a single orientation [2, 3, 4], whereas *bidirected graphs* allow each endpoint to carry its own arrow independently, enabling asymmetric flows on an undirected backbone [5, 6, 7]. *Multidirected graphs* extend this further by permitting multiple parallel arcs between the same pair of vertices, encoding edge multiplicity in directed settings [8]. By leveraging these richer graph models, one can more intuitively visualize and represent real-world systems in which directional information is essential.

Graph Neural Networks (GNNs) have emerged as a powerful paradigm for learning over graph-structured data, achieving state-of-the-art results in social network analysis, recommender systems, and molecular modeling [9, 10]. Recent extensions include Hypergraph Neural Networks [11] and Super-HyperGraph Neural Networks [12], which handle higher-order and hierarchical relations. Research on these GNN architectures plays a vital role across AI, machine learning, reinforcement learning, and many other domains.

1.2 Our Contribution in this paper

Graph Neural Networks (GNNs) and their variants have become indispensable in AI and machine learning due to their strength in modeling relational structures. Directed Graph Neural Networks extend standard GNNs to handle edge orientations, yet architectures specifically designed for bidirected and multidirected graphs remain scarce. Addressing this gap, we introduce two new GNN models tailored to these richer graph types:

- **Bidirected GNNs**, which treat each edge’s direction at its two endpoints separately, propagating incoming and outgoing messages in a distinct manner.
- **Multidirected GNNs**, which incorporate the multiplicity of parallel edges directly into the message-passing and node-update mechanisms.

We present precise mathematical definitions for both architectures, illustrate their operation on representative examples, and show how they effectively capture complex orientation and multiplicity patterns in graph data. Although this paper focuses exclusively on theoretical developments, we anticipate future work involving empirical evaluations on real-world datasets, the design of practical algorithms, and their implementation.

1.3 Structure of This Paper

This paper is organized as follows. In Section 2, we review the definitions of bidirected and multidirected graphs. Section 3 explores Bidirected Graph Neural Networks. Section 4 addresses Multidirected Graph Neural Networks. Finally, Section 5 provides concluding remarks and discusses directions for future work.

2 Preliminaries

In this section we recall the basic definitions and notation used throughout this paper. We assume all graphs and related structures are finite. For standard graph operations and the fundamentals of Graph Neural Networks, the reader may consult the cited references.

2.1 Bidirected and Multidirected Graphs

A *directed graph* (or *digraph*) consists of a finite vertex set V and a set of ordered edges $E \subseteq V \times V$, each edge (u, v) indicating a one-way connection from u (source) to v (target) [13, 14]. A bidirected graph is a generalization of an undirected graph where each edge is assigned a direction at each of its endpoints independently, allowing more expressive edge orientation [7].

Definition 2.1 (Bidirected Graph). [7] A bidirected graph $B = (G, \tau)$ is given by

$$G = (V, E), \quad \tau : V \times E \rightarrow \{-1, 0, 1\},$$

where:

- G is a simple undirected graph on vertex set V and edge set E .
- For each vertex-edge pair (v, e) ,

$$\tau(v, e) = \begin{cases} +1, & \text{if } e \text{ is directed toward } v, \\ -1, & \text{if } e \text{ is directed away from } v, \\ 0, & \text{if } v \text{ is not incident to } e. \end{cases}$$

We call G the underlying graph and τ the bidirection function.

Example 2.2 (Simple Bidirected Path). Let $V = \{A, B, C, D\}$ and

$$G = (V, \{\{A, B\}, \{B, C\}, \{C, D\}\}),$$

with edges $e_1 = \{A, B\}$, $e_2 = \{B, C\}$, $e_3 = \{C, D\}$. Define τ by

$\tau(v, e)$	e_1	e_2	e_3
A	+1	0	0
B	-1	+1	0
C	0	-1	+1
D	0	0	-1

so that each edge is “pointing into” one endpoint and “pointing out of” the other. All other entries are zero.

As a further generalization, one may allow multiple directed edges between the same pair of vertices.

Definition 2.3 (Multidirected Graph). [15, 16, 8] A multidirected graph is a tuple

$$G = (V, E, s, t, m),$$

where

- V is a finite set of vertices,
- E is a finite set of edges,
- $s, t : E \rightarrow V$ assign to each edge its source and target,
- $m : V \times V \rightarrow \mathbb{N}_0$ gives the multiplicity of edges from one vertex to another.

Example 2.4 (MultiDirected Graph). Let $G = (V, E, s, t, m)$ be the MultiDirected Graph defined by

$$\begin{aligned} V &= \{A, B, C\}, \\ E &= \{e_1, e_2, e_3, e_4\}, \\ s(e_1) &= A, \quad t(e_1) = B, & s(e_2) &= A, \quad t(e_2) = B, \\ s(e_3) &= B, \quad t(e_3) = C, & s(e_4) &= C, \quad t(e_4) = A, \end{aligned}$$

with multiplicity function

$$m(u, v) = \begin{cases} 2, & (u, v) = (A, B), \\ 1, & (u, v) = (B, C), \\ 1, & (u, v) = (C, A), \\ 0, & \text{otherwise.} \end{cases}$$

Concretely:

- There are two parallel edges e_1, e_2 from A to B , reflecting $m(A, B) = 2$.
- There is one edge e_3 from B to C , so $m(B, C) = 1$.
- There is one edge e_4 from C back to A , so $m(C, A) = 1$.
- All other ordered pairs of distinct vertices have multiplicity zero.

In tabular form, the multiplicity matrix m is:

$m(u, v)$	$v = A$	$v = B$	$v = C$
$u = A$	0	2	0
$u = B$	0	0	1
$u = C$	1	0	0

This example illustrates a simple three-node MultiDirected Graph with parallel and cyclic edges.

Example 2.5 (Daily Airline Flight Network). Consider a network of three airports:

$$V = \{JFK, LAX, ORD\},$$

and a set of scheduled flights

$$E = \{f_1, f_2, f_3, f_4, f_5\},$$

with source and target functions

$$\begin{aligned} s(f_1) &= JFK, & t(f_1) &= LAX, \\ s(f_2) &= JFK, & t(f_2) &= LAX, \\ s(f_3) &= LAX, & t(f_3) &= JFK, \\ s(f_4) &= LAX, & t(f_4) &= ORD, \\ s(f_5) &= ORD, & t(f_5) &= JFK, \end{aligned}$$

and multiplicity function

$$m(u, v) = \begin{cases} 2, & (u, v) = (JFK, LAX), \\ 1, & (u, v) = (LAX, JFK), \\ 1, & (u, v) = (LAX, ORD), \\ 1, & (u, v) = (ORD, JFK), \\ 0, & \text{otherwise.} \end{cases}$$

Here, there are two distinct daily flights from JFK to LAX, and one flight each on the other listed routes, illustrating a real-world multidirected graph where parallel edges model multiple scheduled connections between the same pair of nodes.

2.2 Directed Graph Neural Networks

Graph Neural Networks (GNNs) iteratively refine node embeddings by propagating and aggregating messages along edges, offering a flexible approach for learning over graph-structured data [12, 17, 18, 19, 20]. The *Directed GNN* (DGNN) extension explicitly separates incoming and outgoing edge messages, ensuring that edge orientation influences each update [21, 22, 23, 24].

Definition 2.6 (Directed Graph Neural Network (DGNN)). (cf.[21, 22, 23, 24]) Let $G = (V, E)$ be a directed graph. Assign each node $v \in V$ an initial feature

$$h_v^{(0)} = x_v \in \mathbb{R}^F,$$

and each directed edge $(u, v) \in E$ an initial embedding

$$e_{uv}^{(0)} \in \mathbb{R}^{F_e}.$$

For layers $\ell = 1, \dots, L$, compute:

$$\begin{aligned} m_v^{\text{in},(\ell)} &= g_{\text{in}}(\{e_{uv}^{(\ell-1)} : (u, v) \in E\}), & m_v^{\text{out},(\ell)} &= g_{\text{out}}(\{e_{vu}^{(\ell-1)} : (v, u) \in E\}), \\ h_v^{(\ell)} &= \phi_v(h_v^{(\ell-1)}, m_v^{\text{in},(\ell)}, m_v^{\text{out},(\ell)}), & e_{uv}^{(\ell)} &= \phi_e(e_{uv}^{(\ell-1)}, h_u^{(\ell)}, h_v^{(\ell)}), \end{aligned}$$

where

$$\mathcal{N}^-(v) = \{u \mid (u, v) \in E\}, \quad \mathcal{N}^+(v) = \{u \mid (v, u) \in E\}.$$

Here, g_{in} and g_{out} are permutation-invariant aggregators (e.g. sum or mean), and ϕ_v, ϕ_e are learnable update functions (e.g. MLPs or GRUs). A final readout

$$h_G = r(\{h_v^{(L)} : v \in V\})$$

yields the graph-level representation, where r is permutation-invariant (e.g. sum- or max-pooling).

Example 2.7 (DGNN on a Four-Node Directed Graph). Let $G = (V, E)$ be the directed graph defined by

$$V = \{A, B, C, D\}, \quad E = \{(A, B), (A, C), (B, D), (C, D)\}.$$

Associate initial node features

$$h_A^{(0)} = x_A, \quad h_B^{(0)} = x_B, \quad h_C^{(0)} = x_C, \quad h_D^{(0)} = x_D,$$

and initial edge features

$$e_{AB}^{(0)} = \alpha, \quad e_{AC}^{(0)} = \beta, \quad e_{BD}^{(0)} = \gamma, \quad e_{CD}^{(0)} = \delta.$$

Choose aggregation functions

$$g_e^- = \sum, \quad g_e^+ = \sum,$$

and let h_v and h_e be learnable MLPs.

At layer $\ell = 1$, for each node $v \in V$ we have incoming and outgoing neighborhoods:

$$\begin{aligned} \mathcal{N}^-(A) &= \emptyset, & \mathcal{N}^+(A) &= \{B, C\}, \\ \mathcal{N}^-(B) &= \{A\}, & \mathcal{N}^+(B) &= \{D\}, \\ \mathcal{N}^-(C) &= \{A\}, & \mathcal{N}^+(C) &= \{D\}, \\ \mathcal{N}^-(D) &= \{B, C\}, & \mathcal{N}^+(D) &= \emptyset. \end{aligned}$$

Therefore, the edge aggregations are

$$\begin{aligned} \bar{e}_A^- &= 0, & \bar{e}_A^+ &= \alpha + \beta, \\ \bar{e}_B^- &= \alpha, & \bar{e}_B^+ &= \gamma, \\ \bar{e}_C^- &= \beta, & \bar{e}_C^+ &= \delta, \\ \bar{e}_D^- &= \gamma + \delta, & \bar{e}_D^+ &= 0, \end{aligned}$$

where summing over an empty set yields the zero vector. The node updates are

$$\begin{aligned} h_A^{(1)} &= h_v(x_A, 0, \alpha + \beta), \\ h_B^{(1)} &= h_v(x_B, \alpha, \gamma), \\ h_C^{(1)} &= h_v(x_C, \beta, \delta), \\ h_D^{(1)} &= h_v(x_D, \gamma + \delta, 0). \end{aligned}$$

Next, each edge feature is updated by

$$\begin{aligned} e_{AB}^{(1)} &= h_e(\alpha, h_A^{(1)}, h_B^{(1)}), \\ e_{AC}^{(1)} &= h_e(\beta, h_A^{(1)}, h_C^{(1)}), \\ e_{BD}^{(1)} &= h_e(\gamma, h_B^{(1)}, h_D^{(1)}), \\ e_{CD}^{(1)} &= h_e(\delta, h_C^{(1)}, h_D^{(1)}). \end{aligned}$$

Subsequent layers $\ell = 2, \dots, L$ proceed analogously. Finally, a global readout

$$h_G = r(\{h_A^{(L)}, h_B^{(L)}, h_C^{(L)}, h_D^{(L)}\})$$

yields the graph-level embedding of G .

3 Result: Bidirected Graph Neural Network

The definition of the Bidirected Graph Neural Network is presented as follows.

Definition 3.1 (Bidirected Graph Neural Network (BiDGNN)). *Let $B = (G, \tau)$ be a bidirected graph with underlying simple graph $G = (V, E)$ and bidirection function $\tau : V \times E \rightarrow \{-1, 0, 1\}$. Associate to each node $v \in V$ an initial feature*

$$h_v^{(0)} = x_v \in \mathbb{R}^F,$$

and to each edge $e \in E$ an initial feature

$$e_e^{(0)} \in \mathbb{R}^{F_e}.$$

For each $v \in V$, define its incoming and outgoing edge sets by

$$\mathcal{E}_\tau^-(v) = \{e \in E : \tau(v, e) = 1\}, \quad \mathcal{E}_\tau^+(v) = \{e \in E : \tau(v, e) = -1\}.$$

A BiDGNN with L layers computes, for $\ell = 1, \dots, L$:

$$\begin{aligned} \bar{e}_v^- &= g_e^-(\{e_e^{(\ell-1)} : e \in \mathcal{E}_\tau^-(v)\}), & \bar{e}_v^+ &= g_e^+(\{e_e^{(\ell-1)} : e \in \mathcal{E}_\tau^+(v)\}), \\ h_v^{(\ell)} &= h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+), & e_e^{(\ell)} &= h_e(e_e^{(\ell-1)}, h_u^{(\ell)}, h_w^{(\ell)}, \tau(u, e), \tau(w, e)), \end{aligned}$$

where $e = (u, w)$, g_e^- and g_e^+ are permutation-invariant aggregators (e.g. sum or mean), and h_v, h_e are learnable update functions (e.g. MLPs or GRUs). A final readout

$$h_B = r(\{h_v^{(L)} : v \in V\})$$

produces a graph-level representation.

Example 3.2 (BiDGNN on a Three-Node Bidirected Graph). Let $B = (G, \tau)$ be the bidirected graph with

$$V = \{A, B, C\}, \quad E = \{e_1, e_2\},$$

where e_1 connects A and B , and e_2 connects B and C . The bidirection function $\tau : V \times E \rightarrow \{-1, 0, 1\}$ is defined by

$$\begin{aligned} \tau(A, e_1) &= 1, & \tau(B, e_1) &= -1, & \tau(v, e) &= 0 \text{ otherwise.} \\ \tau(B, e_2) &= 1, & \tau(C, e_2) &= -1, \end{aligned}$$

Assign initial features

$$h_A^{(0)} = x_A, \quad h_B^{(0)} = x_B, \quad h_C^{(0)} = x_C,$$

and edge features

$$e_{e_1}^{(0)} = \alpha, \quad e_{e_2}^{(0)} = \beta.$$

Choose both aggregators to be sum:

$$g_e^- = g_e^+ = \sum.$$

At layer $\ell = 1$, for each $v \in \{A, B, C\}$ we have

$$\begin{aligned} \mathcal{E}_\tau^-(A) &= \{e_1\}, & \mathcal{E}_\tau^+(A) &= \emptyset, \\ \mathcal{E}_\tau^-(B) &= \{e_2\}, & \mathcal{E}_\tau^+(B) &= \{e_1\}, \\ \mathcal{E}_\tau^-(C) &= \emptyset, & \mathcal{E}_\tau^+(C) &= \{e_2\}. \end{aligned}$$

Thus the aggregated edge messages are

$$\begin{aligned} \bar{e}_A^- &= \alpha, & \bar{e}_A^+ &= 0, \\ \bar{e}_B^- &= \beta, & \bar{e}_B^+ &= \alpha, \\ \bar{e}_C^- &= 0, & \bar{e}_C^+ &= \beta, \end{aligned}$$

where the sum over an empty set is taken to be the zero vector. The node updates are

$$\begin{aligned} h_A^{(1)} &= h_v(x_A, \alpha, 0), \\ h_B^{(1)} &= h_v(x_B, \beta, \alpha), \\ h_C^{(1)} &= h_v(x_C, 0, \beta). \end{aligned}$$

The edge updates, incorporating both endpoint orientations, are

$$\begin{aligned} e_{e_1}^{(1)} &= h_e(\alpha, h_A^{(1)}, h_B^{(1)}, \tau(A, e_1) = 1, \tau(B, e_1) = -1), \\ e_{e_2}^{(1)} &= h_e(\beta, h_B^{(1)}, h_C^{(1)}, \tau(B, e_2) = 1, \tau(C, e_2) = -1). \end{aligned}$$

Subsequent layers proceed similarly. Finally, a global readout

$$h_B = r(\{h_A^{(L)}, h_B^{(L)}, h_C^{(L)}\})$$

yields the graph-level representation of B .

Example 3.3 (BiDGNN on a Four-Node Bidirected Graph). Let $B = (G, \tau)$ be the bidirected graph with

$$V = \{A, B, C, D\}, \quad E = \{e_1, e_2, e_3\},$$

where e_1 connects A and B , e_2 connects A and C , e_3 connects C and D . Define the bidirection function $\tau : V \times E \rightarrow \{-1, 0, 1\}$ by

$$\begin{aligned}\tau(A, e_1) &= -1, & \tau(B, e_1) &= 1, \\ \tau(A, e_2) &= 1, & \tau(C, e_2) &= -1, & \tau(v, e) &= 0 \text{ otherwise.} \\ \tau(C, e_3) &= -1, & \tau(D, e_3) &= 1,\end{aligned}$$

Assign initial node features

$$h_A^{(0)} = x_A, \quad h_B^{(0)} = x_B, \quad h_C^{(0)} = x_C, \quad h_D^{(0)} = x_D,$$

and edge features

$$e_{e_1}^{(0)} = \alpha, \quad e_{e_2}^{(0)} = \beta, \quad e_{e_3}^{(0)} = \gamma.$$

Choose aggregators

$$g_e^- = \text{mean}, \quad g_e^+ = \text{max},$$

where “max” denotes element-wise maximum over the multiset (and both produce the zero vector on empty input). For layer $\ell = 1$, compute for each v :

v	$\mathcal{E}_\tau^-(v)$	$\mathcal{E}_\tau^+(v)$		
A	$\{e_2\}$	$\{e_1\}$		$\bar{e}_A^- = \beta, \quad \bar{e}_A^+ = \alpha,$
B	$\{e_1\}$	$\emptyset$	$\implies$	$\bar{e}_B^- = \alpha, \quad \bar{e}_B^+ = 0,$
C	$\emptyset$	$\{e_2, e_3\}$		$\bar{e}_C^- = 0, \quad \bar{e}_C^+ = \max(\beta, \gamma),$
D	$\{e_3\}$	$\emptyset$		$\bar{e}_D^- = \gamma, \quad \bar{e}_D^+ = 0.$

Node updates:

$$\begin{aligned}h_A^{(1)} &= h_v(x_A, \beta, \alpha), \\ h_B^{(1)} &= h_v(x_B, \alpha, 0), \\ h_C^{(1)} &= h_v(x_C, 0, \max(\beta, \gamma)), \\ h_D^{(1)} &= h_v(x_D, \gamma, 0).\end{aligned}$$

Edge updates (using both endpoint orientations):

$$\begin{aligned}e_{e_1}^{(1)} &= h_e(\alpha, h_A^{(1)}, h_B^{(1)}, -1, 1), \\ e_{e_2}^{(1)} &= h_e(\beta, h_A^{(1)}, h_C^{(1)}, 1, -1), \\ e_{e_3}^{(1)} &= h_e(\gamma, h_C^{(1)}, h_D^{(1)}, -1, 1).\end{aligned}$$

Subsequent layers follow the same pattern of separate incoming/outgoing aggregation and orientation-aware edge updating. Finally, a graph-level readout

$$h_B = r(\{h_A^{(L)}, h_B^{(L)}, h_C^{(L)}, h_D^{(L)}\})$$

yields the embedding of the entire bidirected graph.

Theorem 3.4 (DGNN as a Special Case of BiDGNN). *Let $G_{\text{dir}} = (V, E_{\text{dir}})$ be any directed graph. Construct the bidirected graph*

$$B = (G, \tau), \quad G = (V, E), \quad E = \{\{u, v\} \mid (u, v) \in E_{\text{dir}}\},$$

by setting for each directed edge $(u, v) \in E_{\text{dir}}$ the bidirection signs

$$\tau(v, \{u, v\}) = +1, \quad \tau(u, \{u, v\}) = -1.$$

If we choose the BiDGNN aggregation functions g_e^- and g_e^+ to coincide with the DGNN's in- and out-aggregators, and let the update functions h_v and h_e ignore the extra τ arguments, then the BiDGNN recursion reduces exactly to that of the DGNN on G_{dir} .

Proof. We divide the argument into three steps:

1. Correspondence of edge sets. By construction,

$$E = \{\{u, v\} : (u, v) \in E_{\text{dir}}\},$$

so there is a one-to-one mapping between each directed arc (u, v) and the undirected edge $e = \{u, v\}$.

2. Incoming and outgoing edge-groups. For any vertex $v \in V$, the BiDGNN defines

$$\mathcal{E}_\tau^-(v) = \{e \in E : \tau(v, e) = +1\}, \quad \mathcal{E}_\tau^+(v) = \{e \in E : \tau(v, e) = -1\}.$$

By our choice of τ ,

$$\mathcal{E}_\tau^-(v) = \{\{u, v\} \mid (u, v) \in E_{\text{dir}}\} \quad \text{and} \quad \mathcal{E}_\tau^+(v) = \{\{v, w\} \mid (v, w) \in E_{\text{dir}}\}.$$

Therefore, aggregating over $\mathcal{E}_\tau^-(v)$ exactly matches the DGNN's in-edge set $\{(u, v)\}$, and aggregating over $\mathcal{E}_\tau^+(v)$ matches the DGNN's out-edge set $\{(v, w)\}$.

3. Equivalence of update formulas. The BiDGNN node update at layer ℓ is

$$\begin{aligned} \bar{e}_v^- &= g_e^-(\{e_e^{(\ell-1)} : e \in \mathcal{E}_\tau^-(v)\}), \quad \bar{e}_v^+ = g_e^+(\{e_e^{(\ell-1)} : e \in \mathcal{E}_\tau^+(v)\}), \\ h_v^{(\ell)} &= h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+). \end{aligned}$$

Since $g_e^- = g_{\text{in}}$, $g_e^+ = g_{\text{out}}$, and h_v ignores τ , this is identical to the DGNN's

$$\begin{aligned} m_v^{\text{in},(\ell)} &= g_{\text{in}}(\{e_{uv}^{(\ell-1)} : (u, v) \in E_{\text{dir}}\}), \quad m_v^{\text{out},(\ell)} = g_{\text{out}}(\{e_{vu}^{(\ell-1)} : (v, u) \in E_{\text{dir}}\}), \\ h_v^{(\ell)} &= \phi_v(h_v^{(\ell-1)}, m_v^{\text{in},(\ell)}, m_v^{\text{out},(\ell)}). \end{aligned}$$

Similarly, the BiDGNN edge update

$$e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_u^{(\ell)}, h_v^{(\ell)}, \tau(u, e), \tau(v, e))$$

reduces to

$$e_{uv}^{(\ell)} = \phi_e(e_{uv}^{(\ell-1)}, h_u^{(\ell)}, h_v^{(\ell)})$$

once h_e ignores the τ inputs. Hence all layer-wise equations coincide, proving the claim. $\square$

Theorem 3.5 (Preservation of Bidirected Structure). *The BiDGNN architecture strictly enforces the bidirected labels τ at each layer:*

- Node updates aggregate only those edge features whose local sign $\tau(v, e)$ indicates “incoming” (+1) or “outgoing” (−1).
- Edge updates receive both endpoint signs $(\tau(u, e), \tau(w, e))$, ensuring that the orientation at each end fully influences the new edge embedding.

Proof. We verify the two key invariants of BiDGNN—node-level filtering and edge-level encoding—by direct inspection of the update formulas.

1. Node-level orientation filtering. At layer ℓ , each node v computes

$$\bar{e}_v^- = g_e^-\left(\{e_e^{(\ell-1)} \mid e \in \mathcal{E}_\tau^-(v)\}\right), \quad \bar{e}_v^+ = g_e^+\left(\{e_e^{(\ell-1)} \mid e \in \mathcal{E}_\tau^+(v)\}\right),$$

where

$$\mathcal{E}_\tau^-(v) = \{e \mid \tau(v, e) = +1\}, \quad \mathcal{E}_\tau^+(v) = \{e \mid \tau(v, e) = -1\}.$$

By definition, $\mathcal{E}_\tau^-(v)$ contains exactly those edges directed into v , and $\mathcal{E}_\tau^+(v)$ those directed out of v . Consequently:

- No “incoming” aggregate $\bar{e}_v^-$ ever includes an edge for which $\tau(v, e) = -1$.
- No “outgoing” aggregate $\bar{e}_v^+$ ever includes an edge for which $\tau(v, e) = +1$.

The node-update

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+)$$

thus uses strictly the correctly oriented messages, ensuring that edges whose local orientation does not match the designated role are never mixed into the update.

2. Edge-level orientation encoding. For each edge $e = (u, w)$, the BiDGNN edge update is

$$e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_u^{(\ell)}, h_w^{(\ell)}, \tau(u, e), \tau(w, e)).$$

Here, the pair $(\tau(u, e), \tau(w, e)) \in \{+1, -1\}^2$ is passed explicitly into the learnable function h_e . Thus:

- Any change in the orientation signs $\tau(u, e)$ or $\tau(w, e)$ will directly affect the subsequent embedding $e_e^{(\ell)}$.
- The model can learn to treat edges differently depending on whether they are “pointing in” or “pointing out” at each endpoint.

Combining these two observations, we see that at every layer, BiDGNN (i) filters node messages according to the correct local orientation, and (ii) incorporates both endpoint signs when updating edges. Therefore, the bidirected structure τ is preserved exactly throughout the network. $\square$

Theorem 3.6 (Isomorphism Invariance). *Let $B = (G, \tau)$ and $B' = (G', \tau')$ be two bidirected graphs, and let $\varphi : V(G) \rightarrow V(G')$ be a graph isomorphism such that $\{\varphi(u), \varphi(w)\} \in E(G')$ whenever $\{u, w\} \in E(G)$, and*

$$\tau'(\varphi(v), \varphi(e)) = \tau(v, e) \quad \text{for all } v \in V(G), e \in E(G).$$

If node and edge features are carried over by $h'_{\varphi(v)} = h_v^{(0)}$ and $e'_{\varphi(e)} = e_e^{(0)}$, then after L layers of a BiDGNN with tied parameters,

$$h'_{\varphi(v)}^{(\ell)} = h_v^{(\ell)}, \quad e'_{\varphi(e)}^{(\ell)} = e_e^{(\ell)}, \quad \text{for all } \ell = 0, \dots, L.$$

In particular, the final graph-level readout satisfies $h_{B'} = h_B$.

Proof. We proceed by induction on ℓ . The base case $\ell = 0$ holds by assumption. Suppose the claim holds for $\ell - 1$; then for layer ℓ :

$$\mathcal{E}_\tau^\pm(v) \longleftrightarrow \mathcal{E}_{\tau'}^\pm(\varphi(v)),$$

and since $g_e^\pm$, h_v , h_e are applied identically under the correspondence and only depend on multisets of input features and the values of τ , the updates

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+), \quad e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_u^{(\ell)}, h_w^{(\ell)}, \tau(u, e), \tau(w, e))$$

carry over to

$$h'_{\varphi(v)}^{(\ell)} = h_v^{(\ell)}, \quad e'_{\varphi(e)}^{(\ell)} = e_e^{(\ell)}.$$

By induction, the statement holds for all layers, and permutation-invariance of the readout r then gives $h_{B'} = h_B$. $\square$

Theorem 3.7 (Reduction to Undirected MPNN). *Let $B = (G, \tau)$ be any bidirected representation of an undirected graph $G = (V, E)$, with $\tau(v, e) \in \{\pm 1\}$ assigning opposite signs at the two endpoints of each e . If one chooses*

$$g_e^- = g_e^+ = \text{Sum}, \quad h_e \text{ independent of } \tau, \quad h_v \text{ symmetric in its second and third arguments},$$

then the BiDGNN reduces to a standard message-passing GNN (MPNN) on G :

$$\bar{e}_v = \sum_{e \ni v} e_e^{(\ell-1)}, \quad h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v, \bar{e}_v), \quad e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_u^{(\ell)}, h_w^{(\ell)}).$$

Proof. Since each undirected edge $e = \{u, w\}$ satisfies $\tau(u, e) = 1$, $\tau(w, e) = -1$, we have $\mathcal{E}_\tau^-(v) \cup \mathcal{E}_\tau^+(v) = \{e : v \in e\}$ and these two sets are disjoint. With $g_e^- = g_e^+ = \text{Sum}$, $\bar{e}_v^- + \bar{e}_v^+ = \sum_{e \ni v} e_e^{(\ell-1)}$. Writing $\bar{e}_v = \bar{e}_v^- + \bar{e}_v^+$, and using the symmetry of h_v in its second and third slots,

$$h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+) = h_v(h_v^{(\ell-1)}, \bar{e}_v, \bar{e}_v),$$

so the node update matches that of a standard MPNN. The edge update with h_e ignoring τ likewise coincides. Hence the BiDGNN recovers exactly the usual MPNN on G . $\square$

Theorem 3.8 (Preservation of Bidirected Semantics). *In every layer of a BiDGNN, the separation between incoming and outgoing edge messages at each node is strictly maintained, and edge updates explicitly depend on the orientation at both endpoints. As a result, no information is propagated against the local orientation, and both signs $\tau(u, e)$ and $\tau(w, e)$ are incorporated in each edge's embedding.*

Proof. We prove the claim by examining the node-level and edge-level updates, and then argue by induction that these properties hold at all layers.

1. Node-level message separation. At layer ℓ , each node v computes two separate aggregates:

$$\bar{e}_v^- = g_e^-(\{e_e^{(\ell-1)} \mid \tau(v, e) = +1\}), \quad \bar{e}_v^+ = g_e^+(\{e_e^{(\ell-1)} \mid \tau(v, e) = -1\}).$$

By definition of τ :

- $\bar{e}_v^-$ collects exactly those edges with local orientation “incoming” to v ($\tau(v, e) = +1$).
- $\bar{e}_v^+$ collects exactly those edges with local orientation “outgoing” from v ($\tau(v, e) = -1$).

Hence, the node update

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+)$$

never intermixes messages of opposite orientation: each message enters the appropriate slot.

2. Edge-level orientation encoding. For an edge $e = (u, w)$, the update rule is

$$e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_u^{(\ell)}, h_w^{(\ell)}, \tau(u, e), \tau(w, e)).$$

Since the pair $(\tau(u, e), \tau(w, e)) \in \{+1, -1\}^2$ is supplied to h_e , any change in the local signs will directly affect the new edge embedding. Thus each edge update encodes the bidirected information of both endpoints.

3. Inductive step. Suppose at layer $\ell - 1$, all node embeddings $h_v^{(\ell-1)}$ and edge embeddings $e_e^{(\ell-1)}$ respect the above invariants. Then:

- The aggregates $\bar{e}_v^\pm$ at layer ℓ are formed from correctly oriented features $e_e^{(\ell-1)}$, so the node updates preserve message separation.
- The edge update at layer ℓ again receives both endpoint signs and the updated node features $h_u^{(\ell)}, h_w^{(\ell)}$, ensuring orientation encoding continues.

By induction, these properties hold for every layer $1, \dots, L$.

Therefore, BiDGNN never mixes incoming and outgoing information incorrectly, and always incorporates both endpoint orientations in its edge updates, preserving the bidirected semantics layer after layer. $\square$

4 Result: Multidirected Graph Neural Network

An MDGNN extends GNNs to graphs with multiple directed edges, separately aggregating incoming and outgoing messages based on edge multiplicity. The definition of the Multidirected Graph Neural Network is presented as follows.

Definition 4.1 (MultiDirected Graph Neural Network (MDGNN)). *Let $G = (V, E, s, t, m)$ be a MultiDirected Graph as in [8, 15], where*

- V is the set of vertices,
- E is the set of directed edges,
- $s, t : E \rightarrow V$ assign each edge its source and target,
- $m : V \times V \rightarrow \mathbb{N}_0$ gives the multiplicity of edges between any ordered pair.

Associate each node $v \in V$ with an initial feature

$$h_v^{(0)} = x_v \in \mathbb{R}^F,$$

and each edge $e \in E$ with an initial feature

$$e_e^{(0)} \in \mathbb{R}^{F_e}.$$

For each $v \in V$, denote its incoming and outgoing edge sets by

$$E_v^- = \{e \in E : t(e) = v\}, \quad E_v^+ = \{e \in E : s(e) = v\}.$$

A MDGNN with L layers then performs, for $\ell = 1, \dots, L$:

$$\begin{aligned} \bar{e}_v^- &= g_e^-(\{e_e^{(\ell-1)} : e \in E_v^-\}), & \bar{e}_v^+ &= g_e^+(\{e_e^{(\ell-1)} : e \in E_v^+\}), \\ h_v^{(\ell)} &= h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+), & e_e^{(\ell)} &= h_e(e_e^{(\ell-1)}, h_{s(e)}^{(\ell)}, h_{t(e)}^{(\ell)}, m(s(e), t(e))), \end{aligned}$$

where

- g_e^- and g_e^+ are permutation-invariant multiset aggregators (e.g. sum, mean, or attention),
- $h_v : \mathbb{R}^F \times \mathbb{R}^{F_e'} \times \mathbb{R}^{F_e'} \rightarrow \mathbb{R}^F$ and $h_e : \mathbb{R}^{F_e} \times \mathbb{R}^F \times \mathbb{R}^F \times \mathbb{N}_0 \rightarrow \mathbb{R}^{F_e}$ are learnable update functions (e.g. MLPs or GRUs).

Finally, a global readout

$$h_G = r(\{h_v^{(L)} : v \in V\})$$

produces a graph-level embedding, where r is permutation-invariant (e.g. sum-pooling, max-pooling or set-attention).

Example 4.2 (MDGNN on a Two-Node MultiDirected Graph). Consider the MultiDirected Graph

$$G = (\{A, B\}, E, s, t, m)$$

with

$$E = \{e_1, e_2, e_3\}, \quad s(e_1) = A, t(e_1) = B, \quad s(e_2) = A, t(e_2) = B, \quad s(e_3) = B, t(e_3) = A,$$

and multiplicity

$$m(A, B) = 2, \quad m(B, A) = 1, \quad m(A, A) = m(B, B) = 0.$$

We take initial features

$$h_A^{(0)} = x_A, \quad h_B^{(0)} = x_B, \quad e_{e_1}^{(0)} = \alpha_1, \quad e_{e_2}^{(0)} = \alpha_2, \quad e_{e_3}^{(0)} = \beta.$$

Let both aggregators be sum: $g_e^- = g_e^+ = \sum$. Then for the first layer ($\ell = 1$):

$$\begin{aligned} E_A^- &= \{e_3\}, & E_A^+ &= \{e_1, e_2\}, \\ E_B^- &= \{e_1, e_2\}, & E_B^+ &= \{e_3\}, \end{aligned}$$

$$\bar{e}_A^- = \beta, \quad \bar{e}_A^+ = \alpha_1 + \alpha_2, \quad \bar{e}_B^- = \alpha_1 + \alpha_2, \quad \bar{e}_B^+ = \beta.$$

Node updates:

$$h_A^{(1)} = h_v(x_A, \beta, \alpha_1 + \alpha_2), \quad h_B^{(1)} = h_v(x_B, \alpha_1 + \alpha_2, \beta).$$

Edge updates (using multiplicity):

$$\begin{aligned} e_{e_1}^{(1)} &= h_e(\alpha_1, h_A^{(1)}, h_B^{(1)}, m(A, B) = 2), \\ e_{e_2}^{(1)} &= h_e(\alpha_2, h_A^{(1)}, h_B^{(1)}, m(A, B) = 2), \\ e_{e_3}^{(1)} &= h_e(\beta, h_B^{(1)}, h_A^{(1)}, m(B, A) = 1). \end{aligned}$$

Subsequent layers proceed similarly. Finally, a global readout

$$h_G = r(\{h_A^{(L)}, h_B^{(L)}\})$$

produces the graph-level embedding.

Example 4.3 (MDGNN on a Three-Node MultiDirected Graph with Mixed Multiplicities). *Consider the MultiDirected Graph*

$$G = (\{A, B, C\}, E, s, t, m)$$

with

$$\begin{aligned} E &= \{e_1, e_2, e_3, e_4, e_5\}, \\ s(e_1) &= A, t(e_1) = B, \quad s(e_2) = A, t(e_2) = B, \\ s(e_3) &= B, t(e_3) = C, \quad s(e_4) = C, t(e_4) = A, \\ s(e_5) &= C, t(e_5) = A, \end{aligned}$$

and multiplicities

$$m(A, B) = 2, \quad m(B, C) = 1, \quad m(C, A) = 2, \quad m(x, y) = 0 \text{ otherwise.}$$

Assign initial node features

$$h_A^{(0)} = x_A, \quad h_B^{(0)} = x_B, \quad h_C^{(0)} = x_C,$$

and initial edge features

$$e_{e_1}^{(0)} = \alpha_1, \quad e_{e_2}^{(0)} = \alpha_2, \quad e_{e_3}^{(0)} = \beta, \quad e_{e_4}^{(0)} = \gamma_1, \quad e_{e_5}^{(0)} = \gamma_2.$$

Let the aggregators be

$$g_e^- = \text{mean}, \quad g_e^+ = \sum,$$

and denote for each $v \in \{A, B, C\}$

$$E_v^- = \{e : t(e) = v\}, \quad E_v^+ = \{e : s(e) = v\}.$$

At layer $\ell = 1$, we compute:

$$\begin{aligned} E_A^- &= \{e_4, e_5\}, & E_A^+ &= \{e_1, e_2\}, \\ E_B^- &= \{e_1, e_2\}, & E_B^+ &= \{e_3\}, \\ E_C^- &= \{e_3\}, & E_C^+ &= \{e_4, e_5\}. \end{aligned}$$

Thus the aggregated edge features are

$$\begin{aligned} \bar{e}_A^- &= \text{mean}\{\gamma_1, \gamma_2\}, & \bar{e}_A^+ &= \alpha_1 + \alpha_2, \\ \bar{e}_B^- &= \text{mean}\{\alpha_1, \alpha_2\}, & \bar{e}_B^+ &= \beta, \\ \bar{e}_C^- &= \beta, & \bar{e}_C^+ &= \gamma_1 + \gamma_2. \end{aligned}$$

Node updates:

$$\begin{aligned} h_A^{(1)} &= h_v(x_A, \bar{e}_A^-, \bar{e}_A^+) = h_v(x_A, \frac{\gamma_1 + \gamma_2}{2}, \alpha_1 + \alpha_2), \\ h_B^{(1)} &= h_v(x_B, \bar{e}_B^-, \bar{e}_B^+) = h_v(x_B, \frac{\alpha_1 + \alpha_2}{2}, \beta), \\ h_C^{(1)} &= h_v(x_C, \bar{e}_C^-, \bar{e}_C^+) = h_v(x_C, \beta, \gamma_1 + \gamma_2). \end{aligned}$$

Edge updates (incorporating multiplicity):

$$\begin{aligned} e_{e_1}^{(1)} &= h_e(\alpha_1, h_A^{(1)}, h_B^{(1)}, m(A, B) = 2), \\ e_{e_2}^{(1)} &= h_e(\alpha_2, h_A^{(1)}, h_B^{(1)}, m(A, B) = 2), \\ e_{e_3}^{(1)} &= h_e(\beta, h_B^{(1)}, h_C^{(1)}, m(B, C) = 1), \\ e_{e_4}^{(1)} &= h_e(\gamma_1, h_C^{(1)}, h_A^{(1)}, m(C, A) = 2), \\ e_{e_5}^{(1)} &= h_e(\gamma_2, h_C^{(1)}, h_A^{(1)}, m(C, A) = 2). \end{aligned}$$

Further layers proceed analogously, updating node and edge features by aggregating according to incoming/outgoing sets and using the multiplicity in each edge update. Finally, a graph-level readout

$$h_G = r(\{h_A^{(L)}, h_B^{(L)}, h_C^{(L)}\})$$

yields the embedding for the entire MultiDirected Graph.

Theorem 4.4 (DGNN as a Special Case). *Let $G_{\text{dir}} = (V, E_{\text{dir}})$ be any directed graph. Define*

$$G = (V, E_{\text{dir}}, s, t, m) \quad \text{with} \quad m(u, v) = \begin{cases} 1, & (u, v) \in E_{\text{dir}}, \\ 0, & \text{otherwise.} \end{cases}$$

If one chooses h_e independent of its last argument $m(s(e), t(e))$, and uses the same aggregators $g_e^\pm$, then the MDGNN on G coincides exactly with the DGNN on G_{dir} .

Theorem 4.5 (DGNN as a Special Case of MDGNN). *Let $G_{\text{dir}} = (V, E_{\text{dir}})$ be a directed graph, and construct the multidirected graph*

$$G = (V, E, s, t, m), \quad E = E_{\text{dir}}, \quad s(u, v) = u, \quad t(u, v) = v, \quad m(u, v) = \begin{cases} 1, & (u, v) \in E_{\text{dir}}, \\ 0, & \text{otherwise.} \end{cases}$$

If the MDGNN uses the same node- and edge-aggregators $g_e^\pm$ as the DGNN, and its edge-update function h_e ignores the multiplicity argument $m(s(e), t(e))$, then the MDGNN on G reproduces exactly the DGNN on G_{dir} .

Proof. We show layer-by-layer equivalence by comparing the MDGNN's update rules to those of the DGNN.

1. Incoming and outgoing edge sets coincide. By definition of G ,

$$E_v^- = \{e \in E : t(e) = v\} = \{(u, v) \mid (u, v) \in E_{\text{dir}}\} = \{\text{incoming arcs to } v\},$$

$$E_v^+ = \{e \in E : s(e) = v\} = \{(v, w) \mid (v, w) \in E_{\text{dir}}\} = \{\text{outgoing arcs from } v\}.$$

Thus the MDGNN's in- and out-edge multisets match exactly those of the DGNN.

2. Node update matches. The MDGNN aggregates

$$\bar{e}_v^- = g_e^-(\{e_e^{(\ell-1)} : e \in E_v^-\}), \quad \bar{e}_v^+ = g_e^+(\{e_e^{(\ell-1)} : e \in E_v^+\}),$$

and then updates

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+).$$

Since $g_e^- = g_{\text{in}}$, $g_e^+ = g_{\text{out}}$, and h_v is identical in both models, this coincides with the DGNN's

$$m_v^{\text{in}} = g_{\text{in}}(\{e_{uv}^{(\ell-1)} : (u, v) \in E_{\text{dir}}\}), \quad m_v^{\text{out}} = g_{\text{out}}(\{e_{vu}^{(\ell-1)} : (v, u) \in E_{\text{dir}}\}),$$

$$h_v^{(\ell)} = \phi_v(h_v^{(\ell-1)}, m_v^{\text{in}}, m_v^{\text{out}}).$$

3. Edge update matches. The MDGNN edge update reads

$$e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_{s(e)}^{(\ell)}, h_{t(e)}^{(\ell)}, m(s(e), t(e))).$$

Because $m(s(e), t(e)) = 1$ for all $e \in E_{\text{dir}}$ and h_e ignores this argument, we recover exactly the DGNN's edge update

$$e_{uv}^{(\ell)} = \phi_e(e_{uv}^{(\ell-1)}, h_u^{(\ell)}, h_v^{(\ell)}).$$

4. Readout equivalence. Both models employ the same permutation-invariant readout r on $\{h_v^{(L)}\}$, yielding identical graph-level embeddings.

Combining these observations, we conclude that under the stated choices, the MDGNN on G reproduces each layer of the DGNN on G_{dir} , proving the theorem. $\square$

Theorem 4.6 (BiDGNN as a Special Case of MDGNN). *Let $B = (G_{\text{und}}, \tau)$ be a bidirected graph with underlying simple graph*

$$G_{\text{und}} = (V, E_{\text{und}}), \quad \tau : V \times E_{\text{und}} \rightarrow \{+1, -1\}.$$

Construct a multidirected graph

$$G = (V, E, s, t, m), \quad E = \{e_{u \rightarrow v}, e_{v \rightarrow u} \mid \{u, v\} \in E_{\text{und}}\},$$

by setting

$$s(e_{u \rightarrow v}) = u, \quad t(e_{u \rightarrow v}) = v, \quad m(u, v) = 1 \quad \text{for all } u, v.$$

If we choose the MDGNN's edge-update function h_e to ignore the multiplicity argument and to share parameters for each pair $\{u, v\}$ according to the bidirection signs $\tau(u, \{u, v\})$, $\tau(v, \{u, v\})$, then the MDGNN on G reproduces exactly the BiDGNN on B .

Proof. We establish the equivalence by comparing the message-passing steps of MDGNN on G with those of BiDGNN on B .

1. One-to-one correspondence of edges. Each undirected edge $\{u, v\} \in E_{\text{und}}$ gives rise to exactly two directed edges in G : $e_{u \rightarrow v}$ and $e_{v \rightarrow u}$. We will tie the parameters of their update function h_e so that they reflect the original bidirection τ .

2. Matching incoming/outgoing sets. In the MDGNN, for each vertex v we define

$$E_v^- = \{e \in E : t(e) = v\}, \quad E_v^+ = \{e \in E : s(e) = v\}.$$

By construction,

$$E_v^- = \{e_{u \rightarrow v} : \{u, v\} \in E_{\text{und}}\}, \quad E_v^+ = \{e_{v \rightarrow w} : \{v, w\} \in E_{\text{und}}\}.$$

In the BiDGNN on B , the incoming and outgoing edge-sets at v are

$$\mathcal{E}_\tau^-(v) = \{\{u, v\} : \tau(v, \{u, v\}) = +1\}, \quad \mathcal{E}_\tau^+(v) = \{\{v, w\} : \tau(v, \{v, w\}) = -1\}.$$

By tying the MDGNN's h_e parameters for $e_{u \rightarrow v}$ and $e_{v \rightarrow u}$ so that

$$h_e(\dots, \tau(u, \{u, v\}), \tau(v, \{u, v\})) \equiv \begin{cases} \text{incoming update,} & \tau(v, \{u, v\}) = +1, \\ \text{outgoing update,} & \tau(v, \{u, v\}) = -1, \end{cases}$$

we ensure that aggregating over E_v^- in MDGNN reproduces the BiDGNN's "incoming" aggregate $\bar{e}_v^-$, and similarly for E_v^+ .

3. Equivalence of node updates. The MDGNN node update at layer ℓ is

$$\bar{e}_v^- = g_e^-(\{e_e^{(\ell-1)} : e \in E_v^-\}), \quad \bar{e}_v^+ = g_e^+(\{e_e^{(\ell-1)} : e \in E_v^+\}),$$

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+).$$

Since E_v^- and E_v^+ correspond exactly to $\mathcal{E}_\tau^-(v)$ and $\mathcal{E}_\tau^+(v)$, and we have matched the aggregators $g_e^\pm$ and h_v , this replicates the BiDGNN's node update.

4. Equivalence of edge updates. For each directed edge $e_{u \rightarrow v}$, the MDGNN update is

$$e_{u \rightarrow v}^{(\ell)} = h_e(e_{u \rightarrow v}^{(\ell-1)}, h_u^{(\ell)}, h_v^{(\ell)}, m(u, v)).$$

By design, $m(u, v) = 1$ and h_e ignores this argument, but crucially we tie its parameters so that it depends on $\tau(u, \{u, v\}), \tau(v, \{u, v\})$ in the same way as the BiDGNN's edge update

$$e_{\{u, v\}}^{(\ell)} = h_e(e_{\{u, v\}}^{(\ell-1)}, h_u^{(\ell)}, h_v^{(\ell)}, \tau(u, \{u, v\}), \tau(v, \{u, v\})).$$

Hence each MDGNN edge update recovers the corresponding BiDGNN update exactly.

Since both node- and edge-updates align layer by layer, the MDGNN on G reproduces the BiDGNN on B , completing the proof. $\square$

Theorem 4.7 (Preservation of MultiDirected Structure). *In every layer of an MDGNN, the following invariants hold:*

1. Node-level aggregation separates incoming and outgoing edge features via

$$E_v^- = \{e \in E : t(e) = v\}, \quad E_v^+ = \{e \in E : s(e) = v\}.$$

2. Each edge update explicitly incorporates the multiplicity $m(s(e), t(e))$, so parallel edges affect their own representations.

Proof. We prove these invariants by inspecting the MDGNN update rules and then arguing by induction on the layer index ℓ .

1. Node-level separation. At layer ℓ , each node v computes

$$\bar{e}_v^- = g_e^-(\{e_e^{(\ell-1)} : e \in E_v^-\}), \quad \bar{e}_v^+ = g_e^+(\{e_e^{(\ell-1)} : e \in E_v^+\}).$$

By definition:

$$E_v^- = \{e : t(e) = v\} \quad (\text{incoming edges}), \quad E_v^+ = \{e : s(e) = v\} \quad (\text{outgoing edges}).$$

Since g_e^- and g_e^+ aggregate disjoint multisets, no incoming message is mixed into the outgoing slot, and vice versa. The node update

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+)$$

thus uses only correctly classified incoming and outgoing features.

2. Edge-level multiplicity encoding. Each edge e is updated via

$$e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_{s(e)}^{(\ell)}, h_{t(e)}^{(\ell)}, m(s(e), t(e))).$$

Because the scalar $m(s(e), t(e)) \in \mathbb{N}_0$ is passed directly into h_e , the model can learn to scale or modulate the new embedding $e_e^{(\ell)}$ according to the number of parallel edges between the same pair of vertices.

3. Inductive argument. Assume that at layer $\ell - 1$, all node embeddings $h_v^{(\ell-1)}$ and edge embeddings $e_e^{(\ell-1)}$ satisfy the two invariants. Then:

- The sets E_v^- and E_v^+ remain correctly defined, and the aggregators $g_e^\pm$ continue to combine only the intended disjoint subsets.

- The call to h_e at layer ℓ still receives the correct multiplicity $m(s(e), t(e))$, so any learned dependence on parallel-edge count persists.

By induction, both the separation of incoming/outgoing flows and the explicit encoding of multiplicity hold at every layer. $\square$

Theorem 4.8 (Isomorphism Invariance). *Let $G = (V, E, s, t, m)$ and $G' = (V', E', s', t', m')$ be two multidirected graphs, and let $\varphi : V \rightarrow V'$ and $\psi : E \rightarrow E'$ be bijections such that*

$$s'(\psi(e)) = \varphi(s(e)), \quad t'(\psi(e)) = \varphi(t(e)), \quad m'(\varphi(u), \varphi(v)) = m(u, v) \quad \forall u, v \in V, e \in E.$$

Suppose node and edge features satisfy

$$h_{\varphi(v)}^{(0)} = h_v^{(0)}, \quad e_{\psi(e)}^{(0)} = e_e^{(0)},$$

and that the MDGNN uses the same parameter functions $g_e^\pm, h_v, h_e$ on both graphs. Then for every layer $\ell = 0, 1, \dots, L$,

$$h_{\varphi(v)}^{(\ell)} = h_v^{(\ell)}, \quad e_{\psi(e)}^{(\ell)} = e_e^{(\ell)},$$

and the final readout embeddings coincide:

$$h_{G'} = h_G.$$

Proof. By induction on ℓ . The base case $\ell = 0$ holds by feature correspondence. Assume equality holds at layer $\ell - 1$. Since φ and ψ preserve source/target and multiplicity, the incoming/outgoing multisets $\{e_e^{(\ell-1)} : t(e) = v\}$, $\{e_e^{(\ell-1)} : s(e) = v\}$ map exactly to their primed counterparts. Because $g_e^\pm, h_v, h_e$ are applied identically, the updates

$$h_v^{(\ell)} = h_v(h_v^{(\ell-1)}, \bar{e}_v^-, \bar{e}_v^+), \quad e_e^{(\ell)} = h_e(e_e^{(\ell-1)}, h_{s(e)}^{(\ell)}, h_{t(e)}^{(\ell)}, m(s(e), t(e)))$$

carry over under φ, ψ . By induction, all layers agree, and permutation-invariance of the readout yields $h_{G'} = h_G$. $\square$

5 Conclusion and Future Work

In this paper, we have proposed two novel extensions of Graph Neural Networks—*Bidirected GNNs* and *Multidirected GNNs*. Through simple illustrative examples, we have explored their formal definitions and demonstrated how they rigorously capture endpoint-specific orientations and parallel-edge multiplicities. We believe these architectures enable more precise modeling of directionality in graph neural networks.

Looking ahead, we plan to extend this framework to more general relational structures, including:

- Hypergraphs [25, 26, 11, 27],
- Fuzzy graphs [28, 29, 30],
- Intuitionistic Fuzzy Graphs [31, 32],
- Neutrosophic graphs [33, 34, 35],
- Plithogenic graphs [36, 37, 38, 39],
- Superhypergraphs [40, 41, 42, 43, 44].

We also intend to develop benchmark datasets in which multidirectedness is an inherent property, and to create computational tools—such as Python libraries—for practical experimentation with the proposed models.

Acknowledgments

We extend our sincere gratitude to everyone who provided insights, inspiration, and assistance throughout this research. We particularly thank our readers for their interest and acknowledge the authors of the cited works for laying the foundation that made our study possible. We also appreciate the support from individuals and institutions that provided the resources and infrastructure needed to produce and share this paper. Finally, we are grateful to all those who supported us in various ways during this project.

Data Availability

Not applicable.

Competing interests

Author has declared that no competing interests exist.

References

- [1] Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018.
- [2] Takaaki Fujita. Review of rough turian neutrosophic directed graphs and rough pentapartitioned neutrosophic directed graphs. *Neutrosophic Optimization and Intelligent Systems*, 5:48–79, 2025.
- [3] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.
- [4] Gary Chartrand, Heather Jordon, Vincent Vatter, and Ping Zhang. *Graphs & digraphs*. Chapman and Hall/crc, 2024.
- [5] Erling Wei, Wenliang Tang, and Xiaofeng Wang. Flows in 3-edge-connected bidirected graphs. *Frontiers of Mathematics in China*, 6:339–348, 2011.
- [6] Er Ling Wei, Wen Liang Tang, and Dong Ye. Nowhere-zero 15-flow in 3-edge-connected bidirected graphs. *Acta Mathematica Sinica, English Series*, 30(4):649–660, 2014.
- [7] Rui Xu and Cun-Quan Zhang. On flows in bidirected graphs. *Discrete mathematics*, 299(1-3):335–343, 2005.
- [8] Sebastian Pardo-Guerra, Vivek Kurien George, and Gabriel A Silva. On the graph isomorphism completeness of directed and multidirected graphs. *Mathematics*, 13(2):228, 2025.
- [9] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks. *arXiv preprint arXiv:1909.12264*, 2019.
- [10] Lilas Alrahis and Ozgur Sinanoglu. Graph neural networks for hardware vulnerability analysis-can you trust your gnn? In *2023 IEEE 41st VLSI Test Symposium (VTS)*, pages 1–4. IEEE, 2023.
- [11] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.
- [12] Takaaki Fujita and Florentin Smarandache. *Superhypergraph neural networks and plithogenic graph neural networks: Theoretical foundations*. Infinite Study, 2025.
- [13] Ping Zhang and Gary Chartrand. Introduction to graph theory. *Tata McGraw-Hill*, 2:2–1, 2006.
- [14] Frank Gurski, Carolin Rehs, and Jochen Rethmann. Directed path-width of sequence digraphs. In *International Conference on Combinatorial Optimization and Applications*, pages 79–93. Springer, 2018.
- [15] Sebastian Pardo-Guerra, Vivek Kurien George, Vikash Morar, Joshua Roldan, and Gabriel Alex Silva. Extending undirected graph techniques to directed graphs via category theory. *Mathematics*, 12(9):1357, 2024.
- [16] Takaaki Fujita. Extensions of multidirected graphs: Fuzzy, neutrosophic, plithogenic, rough, soft, hypergraph, and superhypergraph variants. *International Journal of Topology*, 2(3):11, 2025.
- [17] Rui Li, Xin Yuan, Mohsen Radfar, Peter Marendy, Wei Ni, Terrence J O’Brien, and Pablo M Casillas-Espinosa. Graph signal processing, graph neural network and graph learning on biological data: a systematic review. *IEEE Reviews in Biomedical Engineering*, 16:109–135, 2021.

- [18] Amer Marwan El-Samman, Incé Amina Husain, Mai Huynh, Stefano De Castro, Brooke Morton, and Stijn De Baerdemacker. Global geometry of chemical graph neural network representations in terms of chemical moieties. *Digital Discovery*, 3(3):544–557, 2024.
- [19] Xiaojun Kang, Xinchuan Li, Hong Yao, Dan Li, Bo Jiang, Xiaoyue Peng, Tiejun Wu, Shihua Qi, and Lijun Dong. Dynamic hypergraph neural networks based on key hyperedges. *Information Sciences*, 616:37–51, 2022.
- [20] Hongmin Cai, Zhixuan Zhou, Defu Yang, Guorong Wu, and Jiazhou Chen. Discovering brain network dysfunction in alzheimer’s disease using brain hypergraph neural network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2023.
- [21] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7912–7921, 2019.
- [22] Yixuan He, Quan Gan, David Wipf, Gesine D Reinert, Junchi Yan, and Mihai Cucuringu. Gnnrank: Learning global rankings from pairwise comparisons via directed graph neural networks. In *international conference on machine learning*, pages 8581–8612. PMLR, 2022.
- [23] Yixuan He, Gesine Reinert, David Wipf, and Mihai Cucuringu. Robust angular synchronization via directed graph neural networks. *arXiv preprint arXiv:2310.05842*, 2023.
- [24] Guo Zhenyu and Zhang Wanhong. An efficient inference schema for gene regulatory networks using directed graph neural networks. In *2023 42nd Chinese Control Conference (CCC)*, pages 6829–6834. IEEE, 2023.
- [25] Claude Berge. *Hypergraphs: combinatorics of finite sets*, volume 45. Elsevier, 1984.
- [26] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 1, 2013.
- [27] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2022.
- [28] TM Nishad, Talal Ali Al-Hawary, and B Mohamed Harif. General fuzzy graphs. *Ratio Mathematica*, 47, 2023.
- [29] Talal Al-Hawary. Complete fuzzy graphs. *International Journal of Mathematical Combinatorics*, 4:26, 2011.
- [30] Azriel Rosenfeld. Fuzzy graphs. In *Fuzzy sets and their applications to cognitive and decision processes*, pages 77–95. Elsevier, 1975.
- [31] R Parvathi, MG Karunambigai, and Krassimir T Atanassov. Operations on intuitionistic fuzzy graphs. In *2009 IEEE international conference on fuzzy systems*, pages 1396–1401. IEEE, 2009.
- [32] M Karunambigai, S Sivasankar, and K Palanivel. Properties of balanced intuitionistic fuzzy graphs. *International Journal of Research in Since*, 1:01–05, 2014.
- [33] Said Broumi, Mohamed Talea, Assia Bakali, and Florentin Smarandache. Single valued neutrosophic graphs. *Journal of New theory*, (10):86–101, 2016.
- [34] Said Broumi, Mohamed Talea, Assia Bakali, Florentin Smarandache, and PK Kishore Kumar. Shortest path problem on single valued neutrosophic graphs. In *2017 international symposium on networks, computers and communications (ISNCC)*, pages 1–6. IEEE, 2017.
- [35] Muhammad Akram, Hafsa M Malik, Sundas Shahzadi, and Florentin Smarandache. Neutrosophic soft rough graphs with application. *Axioms*, 7(1):14, 2018.
- [36] Min Huang, Fenghua Li, et al. Optimizing ai-driven digital resources in vocational english learning using plithogenic n-superhypergraph structures for adaptive content recommendation. *Neutrosophic Sets and Systems*, 88:283–295, 2025.
- [37] WB Vasantha Kandasamy, K Ilanthenral, and Florentin Smarandache. *Plithogenic Graphs*. Infinite Study, 2020.
- [38] Prem Kumar Singh et al. Single-valued plithogenic graph for handling multi-valued attribute data and its context. *Int. J. Neutrosophic Sci*, 15:98–112, 2021.
- [39] Fazeelat Sultana, Muhammad Gulistan, Mumtaz Ali, Naveed Yaqoob, Muhammad Khan, Tabasam Rashid, and Tauseef Ahmed. A study of plithogenic graphs: applications in spreading coronavirus disease (covid-19) globally. *Journal of ambient intelligence and humanized computing*, 14(10):13139–13159, 2023.
- [40] Florentin Smarandache. *Extension of HyperGraph to n-SuperHyperGraph and to Plithogenic n-SuperHyperGraph, and Extension of HyperAlgebra to n-ary (Classical-/Neutro-/Anti-) HyperAlgebra*. Infinite Study, 2020.
- [41] Shouxian Zhu. Neutrosophic n-superhypernetwork: A new approach for evaluating short video communication effectiveness in media convergence. *Neutrosophic Sets and Systems*, 85:1004–1017, 2025.
- [42] Takaaki Fujita. Directed acyclic superhypergraphs (dash): A general framework for hierarchical dependency modeling. *Neutrosophic Knowledge*, 6:72–86, 2025.
- [43] Mohammed Alqahtani. Intuitionistic fuzzy quasi-supergraph integration for social network decision making. *International Journal of Analysis and Applications*, 23:137–137, 2025.
- [44] Florentin Smarandache. *Introduction to the n-SuperHyperGraph-the most general form of graph today*. Infinite Study, 2022.