scientific reports

OPEN

Graph Geometric Algebra networks for graph representation learning

Jianqi Zhong[1,2,3] & Wenming Cao[1,2,3]✉

Graph neural networks (GNNs) have emerged as a prominent approach for capturing graph topology and modeling vertex-to-vertex relationships. They have been widely used in pattern recognition tasks including node and graph label prediction. However, when dealing with graphs from non-Euclidean domains, the relationships, and interdependencies between objects become more complex. Existing GNNs face limitations in handling a large number of model parameters in such complex graphs. To address this, we propose the integration of Geometric Algebra into graph neural networks, enabling the generalization of GNNs within the geometric space to learn geometric embeddings for nodes and graphs. Our proposed Graph Geometric Algebra Network (GGAN) enhances correlations among nodes by leveraging relations within the Geometric Algebra space. This approach reduces model complexity and improves the learning of graph representations. Through extensive experiments on various benchmark datasets, we demonstrate that our models, utilizing the properties of Geometric Algebra operations, outperform state-of-the-art methods in graph classification and semi-supervised node classification tasks. Our theoretical findings are empirically validated, confirming that our model achieves state-of-the-art performance.

Keywords Geometric Algebra, Graph Neural network, Node classification, Graph classification, Feature embedding

Graph embedding is a technique that aims to represent nodes as low-dimensional vectors, capturing their graph position and the underlying structure of their local graph neighborhood[1]. It is a type of embedding algorithm used in graph pre-processing with the goal of making graphs representing learning more efficiently. In recent years, there has been a growing interest in utilizing Graph Neural Networks (GNNs) for learning graph embeddings, leading to a wide range of applications, such as molecular networks[2,3], social networks[4,5], publication networks[6,7], etc. GNNs broadly follow a message-passing aggregation scheme, where each node sends its feature representation to the nodes in its neighborhood nodes, and updates its feature representation by iteratively aggregating the feature representation from the neighboring nodes. Many works have been proposed to improve the message-passing aggregation scheme. For graph-level tasks[8], presented a discriminative structural graph neural network, where the authors designed new aggregation functions to maximize discrimination capacity and learn discriminative graph representations[9]; leveraged masked self-attentional layers to aggregate different weighted features for graph classification[10]; introduced a novel geometric aggregation scheme, known as Geom-GCN, specifically designed to address the limitations of aggregation in graph neural networks; For node-level tasks[11], presented a general inductive framework, leveraging node feature information to efficiently generate node embeddings for unseen data[12]. provided a characterization of all permutation invariant and equivariant linear layers to approximate any neural aggregation network. Empirically, the design of new graph neural networks (GNNs) has mainly been driven by intuition, heuristics, and iterative experimentation, without a comprehensive theoretical understanding of their properties and limitations[13]. To tackle this issue[13], introduced a theoretical framework to analyze the expressive power of GNNs in capturing diverse graph structures. Though GNNs have achieved enormous success on node and graph classification tasks, GNNs on Euclidean graph has high distortion for modeling complex structures like molecular structures and social networks[14]. Moreover, existing GNN models often encounter inefficiencies when dealing with high-dimensional structural data during the inference process through multi-layer networks[13].

Realizing that the most powerful aggregation functions suffer from a dimensionality curse, this paper focuses on the discrimination capacity of aggregation functions to overcome the above problems. Our framework draws inspiration from the work of[13], which highlights the significant relationship between graph neural networks (GNNs) and the Weisfeiler-Lehman (WL) graph isomorphism test[15]. In this paper, we develop a new

[1]Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China. [2]State Key Laboratory of Radio Frequency Heterogeneous Integration, Shenzhen University, Shenzhen 518060, China. [3]College of Electronic and Information Engineering, Shenzhen University, Shenzhen 518060, China. ✉email: wmcao@szu.edu.cn

neighborhood aggregation scheme(Geometric Algebra Graph Neural Network(GGAN)) that learns to represent and distinguish between different graph structures as the WL graph isomorphism test.

In GGAN, we harness the power of Geometric Algebra tools to capture meaningful graph embeddings. Geometric Algebra is a powerful mathematical tool that has demonstrated remarkable success in various applications, including computer vision[16], Signal and Image Processing[17], etc. We present two GA-based GNN models to learn node and graph embeddings within the Geometric algebra space, including G3-GGAN and G4-GGAN. In this geometric algebra space, Geometric Algebra (GA) enables the transformation of multi-dimensional signals into multivectors, allowing for holistic handling within a new multi-dimensional GA space. This approach helps to preserve the correlations among multiple dimensions and prevent information loss. In addition, the input geometric components can be shared in the multiplication process so as to achieve highly expressive calculations through geometric product. This operation can effectively reduce the parameters of the model. Specifically, the output of the model is very sensitive to the input, and any slight change in the geometric input components will obtain a completely different result. Therefore, the potential relationship between each hidden layer and different hidden layers can be better captured, thereby improving the embedding quality. Extensive experiments demonstrate the superior performance of our proposed GGAN over state-of-the-art methods across various benchmark datasets. The key contributions of this paper are outlined as follows:

- We develop a new neighborhood aggregation scheme, Graph Geometric Algebra Network(GGAN), introducing geometric algebra to learn effective graph representation in geometric algebra space and solve high distortion of graph structure in Euclidean space.
- In GGAN, we propose geometric algebra-based networks: G3-GGAN, and G4-GGAN, leveraging the advantage of high-dimension division to learn and aggregate comprehensive graph features.
- We conduct extensive experiments to quantitatively and qualitatively verify that our GGAN outperforms state-of-the-art methods for both node classification and graph classification. works

Related work

Geometric Algebraic and neural network

Geometric algebra, also known as Clifford algebra, expands the domain of neural networks from real numbers to a more complex number domain. For instance, in the field of few-shot classification[18], introduced a metric network based on geometric algebra for cross-domain few-shot classification. In[19], the authors constructed a geometric algebra-based multi-view Interaction network to capture and aggregate motion features for human motion prediction. Wang, et al. present a new type of convolutional neural network (CNN) based on Reduced Geometric Algebra[16]. Based on previous research, Quaternion and Octonion are also widely used in neural networks. In the work of[20], Parcollet, et al. propose quaternion-based neural network(QRNN) and quaternion-based LSTM, which achieve better performances than RNN and LSTM in automatic speech recognition. Zhu, et al.[21] and Wu, et al.[22] propose quaternion-based convolutional neural network (QCNN) and Deep Octonion networks (DONs) for image classification tasks, respectively.

Graph embedding learning

The graph is mainly embodied by nodes (vertex) and relationships (edges), which can be seen as a collection of nodes and edges, and its advantage is to quickly solve complex relationship problems. In general, in graph calculation, the basic data structure expression is: $G = (V, E)$, where V represents node and E represents edge.

Node classification is the use of labeled nodes V in a given graph G to predict the category of unlabeled nodes V', which is also called semi-supervised node classification.

Graph classification aims to classify graph structure data composed of nodes and edges, which does not rely on the attributes of a specific node or a certain edge but starts from the overall structure of the graph. At present, graph classification is mainly used in the field of natural science research, such as drug and molecules analysis. Specifically, Let $G = \{g_i, y_i\} \sum_{i=1}^{n}$ be a set of graph data, where g_i is the graph, and y_i is the corresponding label, the task is to learn an embedding $f_e(g_i)$ for each entire graph g_i to predict its label $\hat{y}_i$.

Recent research advances have made progress in the deep learning field. Many large-scale neural networks have been proposed[9,23–26]in the graph representing learning, which aims to utilize neural network models to map different graphs into different representations in the embedding space. The graph neural network(GNN)[23]extended conventional neural networks to process data represented in graph domains, leading to improved performance compared to earlier approaches. Graph Convolutional Network (GCN)[24] is the promotion of the convolutional neural network (CNN) in the graphs. Hamilton et al.[1] design the GraphSAGE to sample and aggregate node features from the local neighbors of a node, and use these features information to efficiently generate embedding for nodes that have not been seen before.

To make use of complex-valued embeddings, the authors in[27]proposed the concept of complex space in knowledge graph embedding, capturing asymmetric relationships and maintaining the efficiency of dot product operations. Some recent work goes beyond the complex-valued representations and uses more expressive and hypercomplex representations to model the entities and relationships embedded in the knowledge graph. In the work of[28], a novel method called dual quaternion knowledge graph embeddings (DualE) was applied to knowledge graph embeddings to gain a more complete set of relational information. To reduce the distortion of complex graph data, Dai et al.[29] proposed Quaternion Graph Neural Networks (QGNN) extend the existing GCN models into the Quaternion space, enabling more expressive and powerful graph representation learning. They embed nodes and graphs in quaternion space to enhance graph representation quality and decrease model parameters.

Geometric algebra background

Geometric Algebra(GA), known as Clifford Algebra[30], provides a brand-new algebraic structure on high-dimensional vector spaces. In GA space, a new product called geometric product is introduced. For vectors $\boldsymbol{w}$ and $\boldsymbol{z}$ of $\boldsymbol{G}_n$, the geometric product is defined as:

$$\boldsymbol{u} \odot \boldsymbol{v} = \boldsymbol{u} \cdot \boldsymbol{v} + \boldsymbol{u} \wedge \boldsymbol{v}, \tag{1}$$

where $\boldsymbol{u} \cdot \boldsymbol{v}$ denotes the inner product producing a scalar element, $\boldsymbol{u} \wedge \boldsymbol{v}$ denotes the outer product. The outer product in GA represents a new quantity called bivector, which is used for representing an oriented and limited plane. Suppose $\mathbb{G}_n$ is n-dimensional GA with an orthogonal basis of vectors $n\left\{\mathrm{e}_i\right\}, i = 1, \cdots, n$. The e_i vectors are orthogonal, then $\mathrm{e}_i \odot \mathrm{e}_j = \mathrm{e}_i\mathrm{e}_j = \mathrm{e}_i \cdot \mathrm{e}_j + \mathrm{e}_i \wedge \mathrm{e}_j = \mathrm{e}_i \wedge \mathrm{e}_j$, which means that

$$n\left\{1, \left\{\mathrm{e}_i\right\}, \left\{\mathrm{e}_i\mathrm{e}_j\right\}, \cdots, \left\{\mathrm{e}_1\mathrm{e}_2\cdots\mathrm{e}_n\right\}\right\} = n\left\{1, \left\{\mathrm{e}_i\right\}, \left\{\mathrm{e}_{ij}\right\}, \cdots, \left\{\mathrm{e}_{1,2\cdots n}\right\}\right\}. \tag{2}$$

For instance in $\mathbb{G}_3$, suppose $\{\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3\}$ are orthonormal basis in $\mathbb{G}_3$ space. The geometric product is sensitive to the order of the vector basis, which results in anti-commutative product:

$$\mathrm{e}_i\mathrm{e}_j = \mathrm{e}_{ij} = -\mathrm{e}_j\mathrm{e}_i = -\mathrm{e}_{ji}, (i, j = 1, 2, \cdots, n, i \neq j), \tag{3}$$

$$\mathrm{e}_i^2 = -1, i = 1, \cdots, n, \tag{4}$$

$$\mathrm{e}_i\mathrm{e}_{ij} = \mathrm{e}_i\mathrm{e}_i\mathrm{e}_j = \mathrm{e}_j, i, j = 1, \cdots, n, i \neq j, \tag{5}$$

For arbitrary multivector M in $\mathbb{G}_3$, M can be represented as

$$M = a_0 + a_1\mathrm{e}_1 + a_2\mathrm{e}_2 + a_3\mathrm{e}_3 + a_4\mathrm{e}_{12} + a_5\mathrm{e}_{23} + a_6\mathrm{e}_{13} + a_7\mathrm{e}_{123}, \tag{6}$$

where $a_0, a_1, a_2, a_3, a_4, a_5, a_6$ and a_7 are real numbers, e_{12}, e_{23}, e_{13} is the outer product of two arbitrary vectors. In GA, any multivector $M \in \mathbb{G}_n$ is described by

$$M = E_0 + \sum_{1 \leqslant i \leqslant n} E_i(M)\mathrm{e}_i + \sum_{1 \leqslant i < j \leqslant n} E_{ij}(M)\mathrm{e}_{ij} + \cdots + E_{1\cdots n}(M)\mathrm{e}_{1\cdots n}, E(M) \in \mathbb{R}. \tag{7}$$

Methods

In this work, we introduce Graph Geometric Algebra Networks (GGAN), a novel framework that leverages Geometric Algebra (GA) to learn enhanced embeddings for graph-structured data. Our proposed GGAN framework extends traditional Graph Convolutional Networks (GCNs) by incorporating the rich multi-dimensional representations enabled by GA, offering a more expressive model within the GA space. The architecture presented in Figure.1 illustrates the flow and key operations of the Graph Geometric Algebra Network (GGAN). By utilizing GA-based operations, GGAN significantly reduces the model's parameter count, enabling efficient performance on large graphs while simultaneously improving the expressiveness and quality of the learned graph representations.

Graph Geometric Algebra internal representation

In GA space, GA provides an element containing scalars, vectors, bivectors, trivectors, and any other elements created by the geometric product, which we call it multivector. Multivctor can express any dimensional space elements, and, thus, is the key we introduce GA for Graph neural networks. In GA-GNN, we initialize input features in GA space by multivectors representation, which ensures GA aggregation for graphs. Given l-th layer input feature $\boldsymbol{X}_v^{(l)} \in R^{N \times D}$, where we assume D is divisible by geometric algebra dimension 2^n. We splite $\boldsymbol{X}_v^{(l)}$ into 2^n sub-vectors, each of size $D/2^n$, yielding $D/2^n$ dimensional GA features. Consequently, we represent graph features $\boldsymbol{X}_v^{(l)}$ by multivectors(refers to Eq.7), geometric algebra vector $\boldsymbol{H}_v^{(l),G}$, which contains 2^n components as follows:

$$\boldsymbol{H}_v^{(l),G} = f\left(\boldsymbol{X}_v^{(l)}\right) = \boldsymbol{H}_{v,0}^{(l),G} + \boldsymbol{H}_{v,1}^{(l),G} + \boldsymbol{H}_{v,2}^{(l),G}, \ldots, + \boldsymbol{H}_{v,n-1}^{(l),G}, \tag{8}$$

where $\oplus$ denotes concatenation operation in 2^n components.

Graph neural networks

GNNs utilize the graph structure and node features X_v to obtain a representation vector for each individual node, denoted as h_v, as well as for the entire graph, denoted as h_G. Contemporary GNNs adopt a neighborhood aggregation approach, wherein the node's representation is iteratively updated by aggregating the representations of its neighboring nodes. A node's representation encapsulates the structural information from its L-hop network neighborhood. Concretely, the l-th layer of GCL(Graph Convolutional Layer) in[13] can be defined as:

$$\begin{aligned} \mathrm{a}_\mathrm{v}^{(\mathrm{k})} &= \mathrm{AGGREGATE}^{(\mathrm{k})}\left(\left\{\mathrm{h}_\mathrm{u}^{(\mathrm{k}-1)} : \mathrm{u} \in \mathscr{N}(\mathrm{v})\right\}\right), \\ \mathrm{h}_\mathrm{v}^{(\mathrm{k})} &= \mathrm{COMBINE}^{(\mathrm{k})}\left(\mathrm{h}_\mathrm{v}^{(\mathrm{k}-1)}, \mathrm{a}_\mathrm{v}^{(\mathrm{k})}\right), \end{aligned} \tag{9}$$

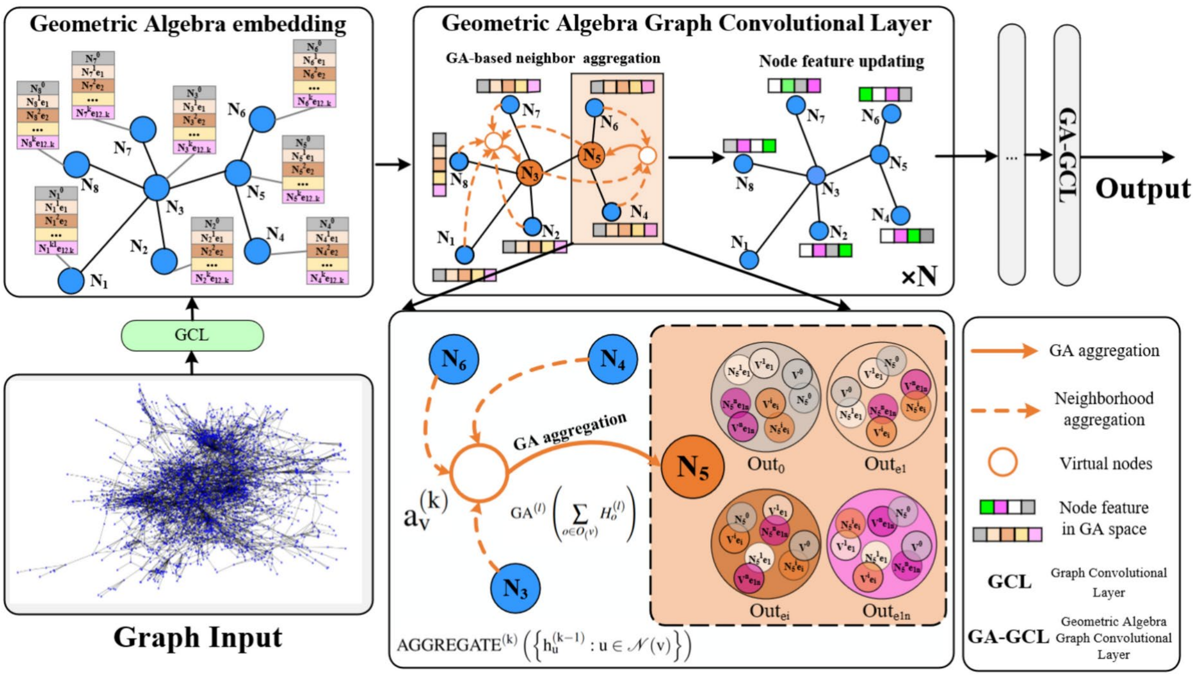
Geometric Algebra embedding
Geometric Algebra Graph Convolutional Layer
GA-based neighbor aggregation
Node feature updating
GA-GCL
Output
×N
GCL
Graph Input
GA aggregation
AGGREGATE^(k)({h_u^(k−1) : u ∈ 𝒩(v)})
GA aggregation
Neighborhood aggregation
Virtual nodes
Node feature in GA space
GCL Graph Convolutional Layer
GA-GCL Geometric Algebra Graph Convolutional Layer

Fig. 1. Overview of Graph Geometric Algebra Network. It is designed to improve the effectiveness of graph neural networks (GNNs) by leveraging Geometric Algebra (GA), which enhances the ability to capture complex relationships in graph data while reducing model complexity.

where $\mathrm{h}_{\mathrm{v}}^{(0)} = \mathrm{X}_v$, $\mathcal{N}(v)$ is a set of nodes adjacent to *v*. Existing works applied different kinds of $\mathrm{AGGREGATE}^{(k)}(\cdot)$ and $\mathrm{COMBINE}^{(k)}(\cdot)$ for graph embedding.

Graph Geometric Algebra network

In this work, we aim to develop a novel architecture, Graph Geometric Algebra Network (GGAN), for graph node updating. Our aim is to achieve the highest level of discriminative power among GNNs, thereby maximizing their effectiveness. Firstly, we achieve feature extraction by first summing up all node features and then introducing geometric algebra to model this process at the node level. Formally, GGAN updates node representations for graph structure as:

$$\mathrm{H}_{\mathrm{v}}^{(l+1),G} = \mathrm{GA}^{(l)}\left(\sum_{o\in O_{(v)}} H_o^{(l)}\right), \tag{10}$$

where $O_{(v)} = \{o \mid (u,v), u \in N(v)\}$, *GA*represents geometric algebra-based operation. For the real-valued graph networks, the vectors are multiplied with the weights using the scalar product. In GGAN, we develop a geometrical algebra-based layer, geometric algebra graph convolutional layer(GA-GCL). The scalar product is replaced by the geometric product for geometric neural networks. Concretely, at the graph level, we follow[13] to integrate AGGREGATE and COMBINE steps(shown in Eq.9) to update graph embeddings as:

$$\begin{aligned}\mathrm{H}_{\mathrm{v}}^{(l+1),G} &= \sigma\left(\sum \mathrm{B}_{\mathrm{u,v}} f\left(\mathrm{w}^{(l)}\right) \otimes f\left(X_v^{(l)}\right)\right), \forall \mathrm{u,v} \in \mathcal{V} \\ &= \sigma\left(\sum \mathrm{B}_{\mathrm{u,v}} \mathrm{W}^{(l,G)} \otimes H_v^{(l,G)}\right), \forall \mathrm{u,v} \in \mathcal{V},\end{aligned} \tag{11}$$

where $\otimes$ is the geometric product in GA space, B represents the adjacency matrix between nodes, $\mathrm{W}^{(l),G}$ is a geometric algebraic weight matrix, and $\mathrm{H}_{\mathrm{v}}^{(l),G}$ is the geometric algebra feature vector of node v at *l*-th layer. $\sigma(\cdot)$ is an activation function such as ELU[31]. $\mathrm{W}^{(l),G}$ and $\mathrm{H}_{\mathrm{v}}^{(l),G}$ can be expressed as follow formula:

$$\mathrm{H}_{\mathrm{v}}^{(l),G} = \mathrm{H}_{\mathrm{v},0}^{(l)} + \mathrm{H}_{\mathrm{v},1}^{(l)}\mathbf{e_1} + \mathrm{H}_{\mathrm{v},2}^{(l)}\mathbf{e_2} + \cdots + \mathrm{H}_{\mathrm{v},2^n-1}^{(l)}\mathbf{e_{12\cdots n}}, k = 2^n, \tag{12}$$

$$\mathrm{W}^{(l),G} = \mathrm{W}_0^{(l)} + \mathrm{W}_1^{(l)}\mathbf{e_1} + \mathrm{W}_2^{(l)}\mathbf{e_2} + \cdots + \mathrm{W}_{2^n-1}^{(l)}\mathbf{e_{12\cdots n}}, k = 2^n. \tag{13}$$

Splitting $\mathrm{H}_{\mathrm{v}}^{(l),G}$ into k parts in the geometric algebraic space, namely, $\mathrm{H}_{\mathrm{v},0}^{(l)}$, $\mathrm{H}_{\mathrm{v},1}^{(l)}$, $\cdots$, $\mathrm{H}_{\mathrm{v},n-1}^{(l)} \in \mathbb{R}$ and corresponding basis vector $\mathbf{e_1}, \mathbf{e_2}, \cdots, \mathbf{e_{12\cdots n}} \in \mathbb{G}_n$, same with $\mathrm{W}^{(l),G}$. We encode the geometric product of $\mathrm{W}^{(l),G}$ and $\mathrm{H}_{\mathrm{v}}^{(l),G}$ as follows:

$$\mathrm{W}^{(l),G} \otimes \mathrm{H}_{\mathrm{v}}^{(l),G} = \begin{bmatrix} 1 \\ \mathrm{e}_1 \\ \mathrm{e}_2 \\ \vdots \\ \mathrm{e}_{12\cdots k} \end{bmatrix}^\top \begin{bmatrix} \mathrm{W}_0^{(l)} & -\mathrm{W}_1^{(l)} & -\mathrm{W}_2^{(l)} & \cdots & -\mathrm{W}_{n-1}^{(l)} \\ \mathrm{W}_1^{(l)} & \mathrm{W}_0^{(l)} & \mathrm{W}_3^{(l)} & \cdots & \vdots \\ \mathrm{W}_2^{(l)} & -\mathrm{W}_3^{(l)} & \ddots & \mathrm{W}_3^{(l)} & \mathrm{W}_2^{(l)} \\ \vdots & \cdots & -\mathrm{W}_3^{(l)} & \mathrm{W}_0^{(l)} & -\mathrm{W}_1^{(l)} \\ \mathrm{W}_{n-1}^{(l)} & \cdots & -\mathrm{W}_2^{(l)} & \mathrm{W}_1^{(l)} & \mathrm{W}_0^{(l)} \end{bmatrix} \begin{bmatrix} \mathrm{H}_{v,0}^{(l)} \\ \mathrm{H}_{v,1}^{(l)} \\ \mathrm{H}_{v,2}^{(l)} \\ \vdots \\ \mathrm{H}_{v,n-1}^{(l)} \end{bmatrix}, \quad (14)$$

where $\mathrm{W}_i^{(l)}$ is shared in the geometric product calculation process. Furthermore, for each $\mathrm{H}_{v,i}^{(l)}$ as long as there is a slight change, the output will be very different, resulting in different performance. This mechanism enables the network to learn the potential relationship between each hidden layer, which effective learning of graph representation to achieve better performance. In this paper, we introduce GGAN in the G^3 and G^4 space, leading to the development of G3-GGAN and G4-GGAN, respectively. These models are designed to learn GA graph embeddings by utilizing real-valued arithmetic to simulate geometric algebra arithmetic, specifically the geometric product.

G3-GGAN

In G3-GGAN, graph features are split into $2^3 = 8$ sub-vectors. Eq.12,13 can be expressed as the following form:

$$\mathrm{H}_{\boldsymbol{v}}^{(l),G} = \boldsymbol{H}_{\mathrm{v},0}^{(l)} + \mathrm{H}_{\mathrm{v},1}^{(l)}\mathbf{e_1} + \mathrm{H}_{\mathrm{v},2}^{(l)}\mathbf{e_2} + \mathrm{H}_{\mathrm{v},3}^{(l)}\mathbf{e_3} + \mathrm{H}_{\mathrm{v},4}^{(l)}\mathbf{e_{12}} + \mathrm{H}_{\mathrm{v},5}^{(l)}\mathbf{e_{31}} + \mathrm{H}_{\mathrm{v},6}^{(l)}\mathbf{e_{23}} + \mathrm{H}_{\mathrm{v},7}^{(l)}\mathbf{e_{123}}, \quad (15)$$

$$\mathrm{W}^{(l),G} = \mathrm{W}_0^{(l)} + \mathrm{W}_1^{(l)}\mathbf{e_1} + \mathrm{W}_2^{(l)}\mathbf{e_2} + \mathrm{W}_3^{(l)}\mathbf{e_3} + \mathrm{W}_4^{(l)}\mathbf{e_{12}} + \mathrm{W}_5^{(l)}\mathbf{e_{31}} + \mathrm{W}_6^{(l)}\mathbf{e_{23}} + \mathrm{W}_7^{(l)}\mathbf{e_{123}}. \quad (16)$$

The geometric product of $\mathrm{W}^{(l),G}$ and $\mathrm{H}_{\mathrm{v}}^{(l),G}$ can be encoded as follows according to Eq.14 and Eq.25:

$$\mathrm{H}_{\mathrm{v}}^{(l),G} \odot \mathrm{W}^{(l),G} = \mathrm{e}^\top \mathrm{M}_v^{\mathrm{G}} \mathrm{H}_v^{(l,G)} = \begin{bmatrix} 1 \\ \mathrm{e}_1 \\ \mathrm{e}_2 \\ \mathrm{e}_2 \\ \mathrm{e}_{12} \\ \mathrm{e}_{31} \\ \mathrm{e}_{23} \\ \mathrm{e}_{123} \end{bmatrix}^\top \begin{bmatrix} \mathrm{W}_0^{(l)} & \mathrm{W}_1^{(l)} & \mathrm{W}_2^{(l)} & \mathrm{W}_3^{(l)} & -\mathrm{W}_4^{(l)} & -\mathrm{W}_5^{(l)} & -\mathrm{W}_6^{(l)} & -\mathrm{W}_7^{(l)} \\ \mathrm{W}_1^{(l)} & \mathrm{W}_0^{(l)} & -\mathrm{W}_4^{(l)} & \mathrm{W}_6^{(l)} & \mathrm{W}_2^{(l)} & -\mathrm{W}_7^{(l)} & -\mathrm{W}_3^{(l)} & -\mathrm{W}_5^{(l)} \\ \mathrm{W}_2^{(l)} & \mathrm{W}_4^{(l)} & \mathrm{W}_0^{(l)} & \mathrm{W}_5^{(l)} & -\mathrm{W}_1^{(l)} & -\mathrm{W}_3^{(l)} & -\mathrm{W}_7^{(l)} & -\mathrm{W}_6^{(l)} \\ \mathrm{W}_3^{(l)} & -\mathrm{W}_6^{(l)} & \mathrm{W}_5^{(l)} & \mathrm{W}_0^{(l)} & -\mathrm{W}_7^{(l)} & -\mathrm{W}_2^{(l)} & \mathrm{W}_1^{(l)} & -\mathrm{W}_4^{(l)} \\ \mathrm{W}_4^{(l)} & \mathrm{W}_2^{(l)} & -\mathrm{W}_1^{(l)} & \mathrm{W}_7^{(l)} & \mathrm{W}_0^{(l)} & -\mathrm{W}_6^{(l)} & \mathrm{W}_5^{(l)} & \mathrm{W}_3^{(l)} \\ \mathrm{W}_5^{(l)} & \mathrm{W}_7^{(l)} & \mathrm{W}_3^{(l)} & -\mathrm{W}_2^{(l)} & \mathrm{W}_6^{(l)} & \mathrm{W}_0^{(l)} & -\mathrm{W}_4^{(l)} & \mathrm{W}_1^{(l)} \\ \mathrm{W}_6^{(l)} & -\mathrm{W}_3^{(l)} & \mathrm{W}_7^{(l)} & \mathrm{W}_1^{(l)} & -\mathrm{W}_5^{(l)} & \mathrm{W}_4^{(l)} & \mathrm{W}_0^{(l)} & \mathrm{W}_2^{(l)} \\ \mathrm{W}_7^{(l)} & \mathrm{W}_5^{(l)} & \mathrm{W}_6^{(l)} & \mathrm{W}_4^{(l)} & \mathrm{W}_3^{(l)} & \mathrm{W}_1^{(l)} & \mathrm{W}_2^{(l)} & \mathrm{W}_0^{(l)} \end{bmatrix} \begin{bmatrix} \mathrm{H}_{v,0}^{(l)} \\ \mathrm{H}_{v,1}^{(l)} \\ \mathrm{H}_{v,2}^{(l)} \\ \mathrm{H}_{v,3}^{(l)} \\ \mathrm{H}_{v,4}^{(l)} \\ \mathrm{H}_{v,5}^{(l)} \\ \mathrm{H}_{v,6}^{(l)} \\ \mathrm{H}_{v,7}^{(l)} \end{bmatrix}. \quad (17)$$

G4-GGAN

In G4-GGAN, graph features are split into $2^4 = 16$ sub-vectors. According to Eq.12,13, $\mathrm{W}^{(l),G}$ and $\mathrm{H}_{\mathrm{v}}^{(l),G}$ can be expressed as:

$$\begin{aligned} \mathrm{H}_{\mathrm{v}}^{(l),G} = & \mathrm{H}_{\mathrm{v},0}^{(l)} + \mathrm{H}_{\mathrm{v},1}^{(l)}\mathbf{e_1} + \mathrm{H}_{\mathrm{v},2}^{(l)}\mathbf{e_2} + \mathrm{H}_{\mathrm{v},3}^{(l)}\mathbf{e_3} + \mathrm{H}_{\mathrm{v},4}^{(l)}\mathbf{e_4} + \mathrm{H}_{\mathrm{v},5}^{(l)}\mathbf{e_{12}} + \mathrm{H}_{\mathrm{v},6}^{(l)}\mathbf{e_{13}} + \\ & \mathrm{H}_{\mathrm{v},7}^{(l)}\mathbf{e_{14}} + \mathrm{H}_{\mathrm{v},8}^{(l)}\mathbf{e_{23}} + \mathrm{H}_{\mathrm{v},9}^{(l)}\mathbf{e_{24}} + \mathrm{H}_{\mathrm{v},10}^{(l)}\mathbf{e_{34}} + \mathrm{H}_{\mathrm{v},11}^{(l)}\mathbf{e_{123}} + \mathrm{H}_{\mathrm{v},12}^{(l)}\mathbf{e_{124}} + \\ & \mathrm{H}_{\mathrm{v},13}^{(l)}\mathbf{e_{134}} + \mathrm{H}_{\mathrm{v},14}^{(l)}\mathbf{e_{234}} + \mathrm{H}_{\mathrm{v},15}^{(l)}\mathbf{e_{1234}}, \end{aligned} \quad (18)$$

$$\begin{aligned} \mathrm{W}^{(l),G} = & \mathrm{W}_0^{(l)} + \mathrm{W}_1^{(l)}\mathbf{e_1} + \mathrm{W}_2^{(l)}\mathbf{e_2} + \mathrm{W}_3^{(l)}\mathbf{e_3} + \mathrm{W}_4^{(l)}\mathbf{e_4} + \mathrm{H}_{\mathrm{v},5}^{(l)}\mathbf{e_{12}} + \mathrm{W}_6^{(l)}\mathbf{e_{13}} + \\ & \mathrm{W}_7^{(l)}\mathbf{e_{14}} + \mathrm{W}_8^{(l)}\mathbf{e_{23}} + \mathrm{W}_9^{(l)}\mathbf{e_{24}} + \mathrm{W}_{10}^{(l)}\mathbf{e_{34}} + \mathrm{W}_{11}^{(l)}\mathbf{e_{123}} + \mathrm{W}_{12}^{(l)}\mathbf{e_{124}} + \\ & \mathrm{W}_{13}^{(l)}\mathbf{e_{134}} + \mathrm{W}_{14}^{(l)}\mathbf{e_{234}} + \mathrm{W}_{15}^{(l)}\mathbf{e_{1234}}. \end{aligned} \quad (19)$$

The geometric product of $\mathrm{W}^{(l),G}$ and $\mathrm{H}_{\mathrm{v}}^{(l),G}$ can be encoded according to Eq.14 and Eq.25.

GA-GNN for graph classification

Based on the proposed GA-GNN(see Figure.1), GA-GNN for graph classification comprises two main layers: GCLs(Graph Convolutional Layers) and GA-GCLs (Geometric Algebra Graph Convolutional Layers). The GCLs ensure that the graph channels meet the operational requirements within the GA space. We stack N GA-GCL layers as the core component to learn graph embeddings. For the graph classification task, the graph structure will be downsized when it is forwarded and finally aggregated into a point feature by multi-layer perceptrons(MLPs). In our work, we follow a powerful and simple Graph Isomorphism Network(GIN)[13] architecture chosen sum pooling as *Readout* function to obtain the embedding E_g of the entire graph G by aggregating the node-level representations:

$$\mathrm{E}_g = \sum_{\mathrm{v}\in\mathscr{V}} \mathrm{E}_v = \sum_{\mathrm{v}\in\mathscr{V}} \left[\mathrm{H}_{\mathrm{v}}^{(1)} \oplus \mathrm{H}_{\mathrm{v}}^{(2)} \oplus, \ldots, \oplus \mathrm{H}_{\mathrm{v}}^{(L)}\right]. \tag{20}$$

The *Readout* function is a critical component in graph neural networks for discriminative tasks. Its main objective is to aggregate the node features from the final layer of the graph and produce a graph-level representation H_g:

$$\mathrm{H}_g = \mathrm{Readout}\left(\left\{\mathrm{H}_v^{(l)}, v \in \mathscr{V}\right\}\right), \tag{21}$$

where $\mathrm{H}_v^{(l)}$ denotes the vector representation of node *v* at the *l*-th layer. We concatenate the vector representations of node v on different layers to construct node embeddings E_{v}:

$$\mathrm{E}_{\mathrm{v}} = \mathrm{H}_{\mathrm{v}}^{(1)} \oplus \mathrm{H}_{\mathrm{v}}^{(2)} \oplus, \ldots, \oplus \mathrm{H}_{\mathrm{v}}^{(L)}, \forall v \in \mathscr{V}. \tag{22}$$

GGAN for node classification
The architecture for node classification shows a similar architecture as graph classification(see Figure.1). The embedding update formula of the multilayer graph neural network can be expressed as follows:

$$\mathrm{H}_{\mathrm{v}}^{(l+1),G} = \sigma\left(\tilde{\mathrm{L}}_{sym} \mathrm{W}^{(l),G} \otimes \mathrm{H}_{\mathrm{v}}^{(l),G}\right), \tag{23}$$

where $\mathrm{L}_{sym} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is a re-normalized Laplacian matrix. It can effectively prevent the disappearance of gradients that occur during multi-layer optimization. In addition, we perform a log-softmax operation on the final output to get the probability distribution of different category labels. In GA-GCL, we introduce geometric algebra space into knowledge graph embedding:

$$\hat{\mathrm{y}}_{\mathrm{v}} = \mathrm{log\_softmax}\left(\sum \tilde{\mathrm{L}}_{sym} \mathrm{W}^{(l,G)} H_v^{(l)}\right), \forall v \in \mathscr{V}. \tag{24}$$

Geometric Algebra embedding generation
The following Algorithm 1 shows the details of how geometric algebra is introduced to embedding generation beyond Euclidean space(we take node classification for instance).

```
Require: B[illegible], $\mathbf{H}_{\mathrm{v}}^{(l),G}$
    Initialize the weight matrix of every GA-GCL layer and normal layer.
```

Algorithm 1. Geometric Algebraic Embedding Generation for graph classification.

Weight initialization strategy

Weight initialization
Weight initialization is a critical component in neural network training, which aims to prevent layer activation outputs from exploding or vanishing during a forward and back pass. Loss gradients may become either too large or too small when the weights are initialized with the same value, which leads to a longer convergence for the network and fewer features the network can learn from training. Therefore, an adequate weight initialization strategy is required to ensure the model converges stably and quickly. In this work, we follow[22] using the Gaussian random method for the GA weigh initialization.

Weight structure
For the selection of the sign(positive or negative) of $\mathrm{W}_i^{(l)}$ in the Equation 14, we use the following formula to explain: Let vector $\boldsymbol{a} = \mathrm{e}_{n_1,n_2,n_3,\cdots,n_k}$, which means $\boldsymbol{a}$ is obtained by multiplying k basis vectors. $\boldsymbol{b} = \mathrm{e}_{m_1,m_2,m_3,\cdots,m_i}$, which means $\boldsymbol{b}$ is obtained by multiplying i basis vectors, where $n_1 < n_2 <, \cdots, < n_k, m_1 < m_2 <, \cdots, m_k$

. Suppose there are s equal basis vectors in the $\boldsymbol{a}$ and $\boldsymbol{b}$ vectors, namely $\mathrm{e}_{(n_{s1})}, \mathrm{e}_{(n_{s2})}, \cdots, \mathrm{e}_{(n_{ss})}$ and $\mathrm{e}_{(m_{s1})}, \mathrm{e}_{(m_{s2})}, \cdots, \mathrm{e}_{(m_{ss})}$. Among them, $n_{si}, m_{si}, i = 1, 2, \cdots, s$ are subscript indexes:

$$p = \mathrm{s} \cdot (\mathrm{n}_{k+1}) - \sum_{i=1}^{S} (n_{si} - m_{si} + i) \,. \tag{25}$$

During the geometric algebraic product, the shared parameters $\mathrm{W}_i^{(l)} (i \in 0, 1, 2 \ldots, n)$ reduce the number of parameter updates during network backpropagation. The output of the geometric algebraic product is sensitive to $\mathrm{h}_{u,i}^{(l)} (i \in 0, 1, 2 \ldots, n)$[29]. This enables the model to learn the underlying relationships between features more effectively and enhance the expressive power of the network.

Experiments

We evaluate the effectiveness of our proposed GGAN(G3-GGAN, G4-GGAN) with State-of-the-Art methods. We conduct extensive experiments for node classification on three public datasets and graph classification on seven well-known datasets which are usually cited by researchers for graph classification. For the comparison of model size, we make a run-time analysis between hypercomplex-based GNN, traditional GNN, and our GA-GNNs given a fixed architecture on a specific dataset.

Experiment baselines

We compared our method with baselines consisting of 15 deep learning methods for graph classification, which include GCAPS[32], GIN[13], DGCNN[33], PPGN[34], CapsGNN[35], DSGC[8], GFN[36], MLC-GCN[37], QGNN[29], CapsualGNN[38], $\pi-$GNN[39], GSN-v[40], ARMA[41], HGRL[42], LPD-GCN[43], LCNN[44]. Five up-to-date baselines(QGNN[29], Geom-GCN[10], N-GCN[45], Graph-MLP[46], EM-GCN[47]) is for node classification. For the aforementioned baseline methods, we report the accuracy values as provided in their respective original papers.

Graph classification

Datasets

There are a wide variety of benchmarks for graph classification. In this work, we use seven well-known datasets consisting of three social network datasets (COLLAB, IMDB-Binary(IMDB-B) and IMDB-Multi(IMDB-M))[48] and four biological datasets (MUTAG, PROTEINS, D&D, NELL, and PTC).

Metrics

To ensure a fair comparison with existing results, we adopt the evaluation methodology used in prior studies[8,29,34]. Specifically, we employ 10-fold cross-validation for graph classification, where the datasets are divided into 10 subsets, with one subset reserved for testing and the remaining subsets used for training. We repeated 10 times for this process and the final performance is reported as the average across the 10 iterations.

Experimental details

For all the experiments, we train our model with 6 GA-GNNS using the proposed G3-GGAN, G4-GGAN. In addition, to illustrate the effectiveness of our models, We conduct additional experiments with varying the number of hidden layers (1, 2, 3, 4, 5) and the hidden size (16 to 128) in our model. The models are trained for a maximum of 150 epochs using the Adam optimizer[51] with a learning rate ranging from $5e_5$ to e_1. Since the social network datasets lack node features, we adopt the approach proposed in[33] where we use node degrees as features for these datasets.

Comparisons

Table. 1 and 2present our comparative results to other methods on both social and biological graphs. In general, we outperform most of the other methods with regards to the task of graph classification, as well as obtain comparable results on datasets of MUTAG. Especially we achieve higher accuracy of 87.29%, 80.80%, 56.00% on the social datasets, COLLAB, IMDA-B, IMDA-M, respectively, which outperforms the state-of-art results[40]by a high margin(2.10%, 3.86%, 3.13%, respectively, and the traditional graphs networks[36] by a margin of 7.10%, 10.68%, 8.11%, respectively.

Our research hypothesis concerning the proposed G3-GGAN, G4-GGAN, is that the increasing dimension number for graph embedding and feature representation in GA space can significantly enhance the performance of the traditional GNNs[13,24,36]. This attributes to certain datasets in our experiments exhibit characteristics that inherently favor higher-dimensional representations. For example, datasets with more intricate graph topology or richer feature information align well with the enhanced modeling capacity of G4-GGAN. Conversely, G3-GGAN's performance may be constrained on such datasets(shown in Table 1,2), especially when the GA dimension is not fully aligned with the data complexity. We confirm it on most datasets except for PTC. For example, the accuracy on PROTEINS obtained by G3-GGAN, G4-GGAN is gradually increasing and outperforms the traditional GNNs by a margin of 4.05%, 6.03%, respectively. However, this rule does not apply to PTC: G3-GGAN yields the best performance on PTC. This perhaps is because the input dimension is adjusted to the size coordinating with GA dimension and graphs, thus losing the embedded parts feature. This would not happen every time but in case the dataset is compared little-size. Overall, the experimental results on both social

Model	MUTAG	PROTEINS	PTC	D&D	NCI1
GCAPS[32]	-	76.40 ± 4.17	66.01 ± 5.91	77.62 ± 4.99	82.72 ± 2.38
GIN[13]	89.40 ± 5.60	76.20 ± 2.80	64.60 ± 7.00	-	82.70 ± 1.70
DGCNN[33]	85.83 ± 1.66	75.54 ± 0.94	58.59 ± 2.47	79.37 ± 0.94	74.44 ± 0.47
PPGN[34]	90.55 ± 8.70	77.20 ± 4.73	66.17 ± 6.54	-	-
CapsGNN[35]	86.67 ± 6.88	76.28 ± 3.63	-	75.38 ± 4.17	78.45 ± 0.26
DSGC[8]	86.70 ± 7.60	74.20 ± 3.80	-	77.40 ± 6.40	79.80 ± 1.20
GFN[36]	90.84 ± 7.22	76.46 ± 4.06	-	78.78 ± 3.49	82.77 ± 1.49
MLC-GCN[37]	86.84	76.52	-	80.86	-
QGNN[29]	92.59 ± 3.59	78.47 ± 3.30	69.92 ± 2.59	79.92 ± 3.54	-
CapsualGNN[38]	93.13	-	-	-	81.2
LPD-GCN[43]	94.80 ± 4.3	-	74.6 ± 4.50	82.90 ± 2.4	82.90 ± 1.50
π-GNN[39]	86.10 ± 8.40	73.60 ± 3.5	-	78.10 ± 3.40	76.70 ± 1.70
GSN-v[40]	92.20 ± 7.50	76.60 ± 5.00	68.20 ± 7.20	-	83.50 ± 2.00
LCNN[44]	**95.73±3.96**	-	73.24 ± 3.35	-	82.05 ± 0.18
ARMA[41]	91.5 ± 4.20	73.70 ± 3.40	-	77.60 ± 2.70	-
DEFL[49]	-	79.46 ± 6.13	-	81.03 ± 5.82	80.03 ± 1.93
HGRL[42]	89.13	78.71	63.59	83.30	85.01
G3-GGAN(Ours)	93.57 ± 5.54	79.70 ± 2.80	**75.86±4.78**	84.46 ± 3.18	82.17 ± 1.64
G4-GGAN(Ours)	94.15±6.35	**81.68±2.87**	72.37 ± 4.62	**85.23±2.86**	**85.86±1.86**

Table 1. Performance results(%) of our methods on biological graphs compared with other baselines. The best scores are in bold.

Model	COLLAB	IMDA-B	IMDA-M	Average
GCAPS[32]	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10	65.97
GIN[13]	80.20 ± 1.90	75.10 ± 5.10	52.30 ± 2.80	69.20
DGCNN[33]	73.61 ± 0.49	70.03 ± 0.86	47.83 ± 0.85	63.82
PPGN[34]	81.38 ± 1.42	73.00 ± 5.77	50.46 ± 3.59	68.28
CapsGNN[35]	79.62 ± 0.91	73.10 ± 4.83	50.27 ± 2.65	67.66
DSGC[8]	79.20 ± 1.60	73.20 ± 4.90	48.50 ± 4.80	66.97
GFN[36]	81.50 ± 2.42	73.00 ± 4.35	51.80 ± 5.16	68.77
MLC-GCN[37]	-	71.4	51.53	-
QGNN[29]	81.36 ± 1.31	77.56 ± 2.45	53.78 ± 3.83	70.90
CapsualGNN[38]	82.33	75.1	52.5	69.98
π-GNN[39]	75.70 ± 1.70	71.50 ± 4.00	48.50 ± 3.50	65.23
G-mixup[50]	-	73.93 ± 2.84	50.29 ± 2.30	-
GSN-v[40]	85.50 ± 1.20	77.80 ± 3.30	54.30 ± 3.30	72.53
G3-GGAN(Ours)	84.87 ± 1.11	75.31 ± 4.88	54.30 ± 3.46	71.49
G4-GGAN(Ours)	**87.29±1.23**	**80.80±3.16**	**56.00±3.77**	**74.70**

Table 2. Performance results(%) of our methods on biological graphs compared with other baselines. The best scores are in bold.

and biological datasets indicate our approaches achieve a competitive performance on all benchmark datasets, which implies representation learning in GA space can be significantly improved.

Node classification

Datasets

The node classification datasets used in this study consist of three widely recognized benchmarks: Cora, Citeseer, and Pubmed[52]. These datasets are citation networks, with each dataset representing a collection of documents. The Cora dataset comprises 2,708 machine learning publications that are categorized into 7 distinct classes. Similarly, the Citeseer dataset consists of 3,327 scientific papers divided into 6 categories. In both Cora and Citeseer, each paper is represented by a one-hot vector indicating the presence of specific words from a dictionary. On the other hand, the Pubmed dataset comprises 19,717 publications related to diabetes. Each paper in the Pubmed dataset is represented by a TF-IDF vector, which captures the importance of terms within the document. In all of these datasets, each node corresponds to a document and is assigned a class label representing the document's main topic. The objective is to predict the correct class for each node in the dataset.

Metrics
To ensure a fair comparison, we adopt the same experimental setup as[29] by using identical 10 random data splits and a 10-fold cross-validation scheme. Each data split is divided into three sets: training, validation, and testing. The proportions of nodes in each class are equally distributed among the sets. Specifically, 60% of the nodes are allocated for training, 20% for validation, and the remaining 20% for testing. This consistent data-splitting strategy allows for reliable and comparable evaluations of different models across multiple experiments.

Experimental details
To illustrate the effectiveness of our models, we compare with the baselines QGNN[29], Geom-GCN[10], N-GCN[45], Graph-MLP[46], EM-GCN[47]. We apply the same architecture as[29], Geom-GCN[10], which is a 2-layer GCN with 16 hidden outputs for CORA, CITESEER, and 64 for PUBMED, but replace the GCL with our proposed GA-GCL. All models to be trained are set to a maximum of 500 epochs using the Adam optimizer[14]. The learning rate is initially set to 0.05, and decay by multiplying 0.8 for every step size epoch of 50 from the 50-th epoch.

Comparisons
Table.3 presents the results of our methods and the state-of-art models on the benchmark datasets. Overall, our methods demonstrate superior performance on larger datasets. Specifically, on the CORA benchmark dataset, our proposed G3-GGAN and G4-GGAN outperform recent works. Notably, G3-GGAN exhibits better performance compared to G4-GGAN (despite having a higher GA dimension), which can be attributed to the same reason explained in Section 2.6. On the second-largest benchmark dataset (CITESEER), G3-GGAN achieves an accuracy of 81.95%, surpassing EM-GCN by 1.67%. For the PUBMED dataset, G3-GGAN outperforms all other approaches.

Interestingly, the application of a low GA dimension in GA-GNN does not yield notable improvements until G4-GGAN achieves a competitive accuracy result (92.21%), surpassing EM-GCN. The occasional underperformance of G3-GGAN relative to baseline models can be explained by its sensitivity to input dimension and graph size. G3-GGAN requires that input features be split into sub-vectors matching the GA dimension ($2^3 = 8$ components in G^3 space). In cases where the input dimensionality or dataset size(e.g., IMDB-M, IMDB-B) is small, this alignment can lead to loss of feature granularity or uneven distribution of information across GA components. This issue is particularly evident in smaller datasets, where G3-GGAN's restricted expressiveness in lower-dimensional GA spaces may limit its ability to outperform baselines with simpler architectures optimized for such settings.

Qualitative Analysis: To visually demonstrate the learning of node embeddings in our proposed GGAN, we utilize t-SNE[53]to visualize the learned node features from the middle layer of our methods, namely Geom-GNN[10]and QGNN[29]. Figure 2 illustrates the clustering effect on the CORA dataset, where points with different colors represent different groups. Comparing the node representation distributions obtained by Geom-GNN and QGNN, we observe that the group centers of each class obtained by G3-GGAN and G4-GGAN are significantly distant from other classes. Moreover, the nodes within each class are closely clustered together, indicating that our methods produce high-quality learned node embeddings.

Ablations study
Model size and computational cost comparisons
To verify the applicability of the proposed GGAN, We compared the computational cost of G3-GGAN and G4-GGAN (our models) with traditional GNNs, including Geom-GNN and QGNN, across different network depths (Layer-1 to Layer-5). The hidden dimension is set to 128. The metrics include model size, running time(ms), and FLOPs(G). We have several observations as follows(see Table 4):

- All models with a 1-layer architecture possess an identical number of parameters. This similarity arises from the models sharing the same output dimension for a given dataset.
- Our models exhibit significantly lower model sizes compared to traditional GNNs, particularly in deeper layers. For instance, at Layer-5, G4-GGAN achieves the smallest size (0.039M), reducing memory requirements by over 84% compared to Geom-GNN(0.250M) and by 69% compared to QGNN (0.125M).
- In terms of running time, our models perform comparably with traditional GNNs for shallow layers (Layer-1 to Layer-3).
- G4-GGAN also requires fewer FLOPs as the network deepens. For example, at Layer-5, it reduces FLOPs by 25% compared to Geom-GNN (42.11G vs. 31.13G).

Model	CORA	CITESEER	PUBMED	Average
Geom-GNN[10]	85.77	73.68	88.13	82.53
N-GCN[45]	83.00	72.20	79.50	78.23
Graph-MLP[46]	79.50	73.10	79.70	77.43
QGNN[29]	87.48 ± 1.08	76.03 ± 1.89	87.65 ± 0.47	83.72
EM-GCN[47]	88.70 ± 0.04	80.60 ± 0.10	91.10 ± 0.10	86.8
G3-GGAN(Ours)	**89.59±1.35**	78.78 ± 1.29	88.98 ± 0.56	85.78
G4-GGAN(Ours)	87.54 ± 1.17	**81.95±1.74**	**92.21±0.56**	**87.23**

Table 3. Graph classification performance(%) of our method are presented. The best are in bold.

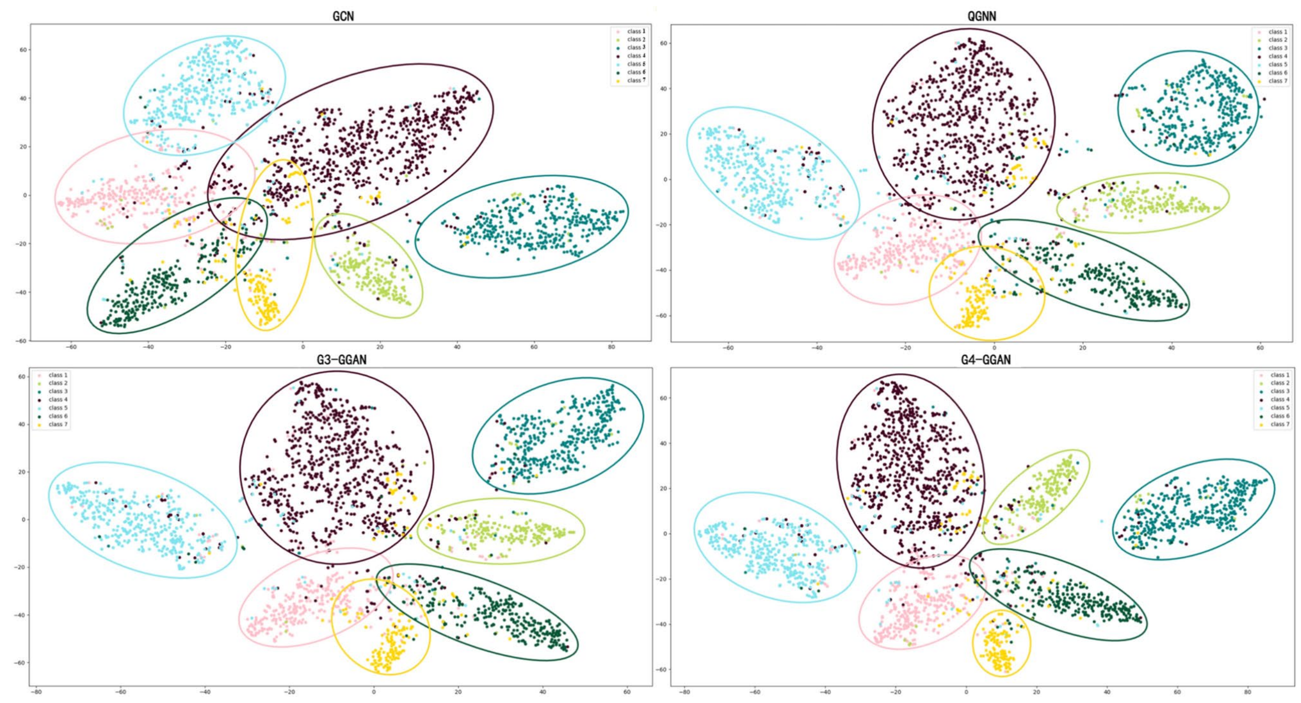

Fig. 2. The figure shows the node representation comparison on dataset CORA. All categories on CORA are represented by different colors. T-SNE methods are used to visualize the node representation performance. We concatenate the components of the G3-GGAN, G4-GGAN node embeddings as the real-valued vector inputs of t-SNE.

	Layer-1			Layer-2			Layer-3		
Methods	**Model size(M)**	**Run time(ms)**	**FLOPs(G)**	**Model size(M)**	**Run time(ms)**	**FLOPs(G)**	**Model size(M)**	**Run time(ms)**	**FLOPs(G)**
Geom-GNN[10]	0.022	10.94	1.25	0.237	11.42	35.12	0.242	14.31	36.48
QGNN[29]	0.022	10.87	1.25	0.119	11.35	31.52	0.121	14.36	34.21
G3-GGAN(Ours)	0.022	10.89	1.25	0.062	11.29	28.58	0.065	14.60	31.52
G4-GGAN(Ours)	0.022	10.93	1.25	0.033	11.95	26.72	0.035	14.35	28.19
	Layer-4			**Layer-5**					
Methods	**Model size(M)**	**Run time(ms)**	**FLOPs(G)**	**Model size(M)**	**Run time(ms)**	**FLOPs(G)**			
Geom-GNN[10]	0.246	16.39	38.85	0.250	22.25	42.11			
QGNN[29]	0.123	16.46	37.25	0.125	22.79	40.58			
G3-GGAN(Ours)	0.071	16.57	33.87	0.074	20.50	36.87			
G4-GGAN(Ours)	0.039	16.04	29.66	0.042	22.36	31.13			

Table 4. Comparison of running time, model size, and computational cost.

We can conclude that GGAN models are computationally efficient, particularly regarding memory usage and FLOPs. The computational performance demonstrates that our GGAN aids in parameter reduction. G3-GGAN and G4-GGAN's compactness is due to their optimized geometric algebra(GA) representation, which achieves high expressiveness with fewer parameters. As the GGAN dimension increases, the parameters decrease due to the inherent weight-sharing mechanism. The compactness of our models ensures scalability to larger graphs without a linear increase in memory or computational requirements. For example, G4-GGAN's smaller model size (0.039M at Layer-5) enables its deployment on large-scale graphs while maintaining efficiency. Additionally, the structure of GA-based models allows for the natural partitioning of graph data. This facilitates parallel processing and distributed computation, critical for handling larger graphs.

The effect of model depth

To examine the impact of model layer depth (number of layers) on classification performance, we evaluate the results on CORA dataset using QGNN[29], G3-GGAN, G4-GGAN, and the standard Geom-GNN[10]models. We vary the number of layers from 1 to 5 in our experiments. We follow the experimental setting of[29]for QGNN and[10] for GCN. The hyperparameters are chosen as follows: 0.5 for the dropout rate, 500 epochs for training

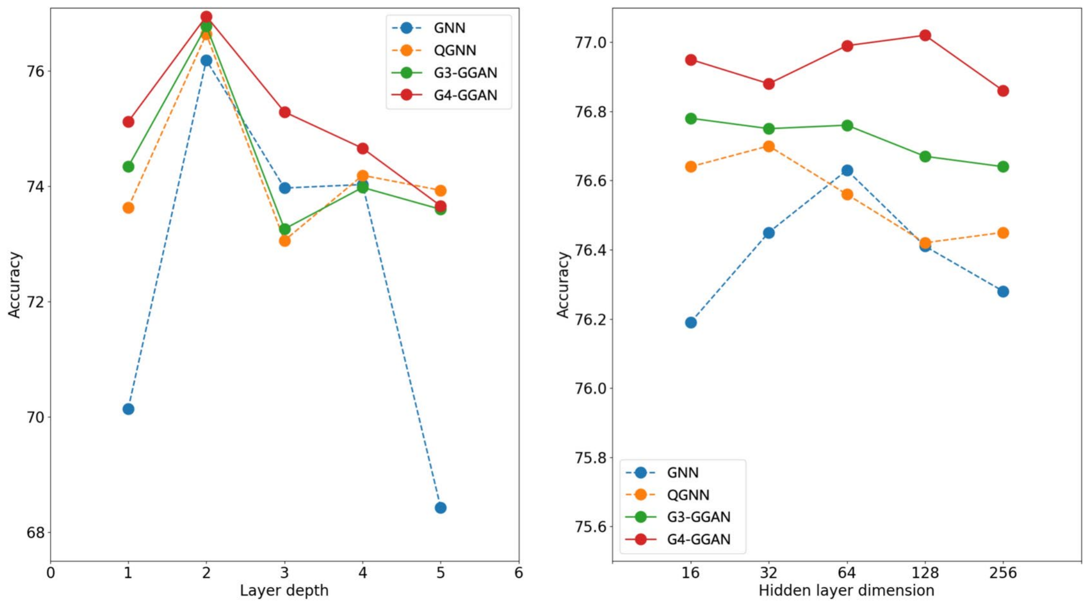

Fig. 3. node classification results comparison on CITESEER by investigating the influence of model depth(shown in (a)) and hidden layer(shown in (b)) dimension to model performance.

using the Adam optimizer, 0.05 for the learning rate, and 16 for the number of the hidden layer. Fig 3(a) reports the result.

We observe that the performance of all models on classification reaches the peak when model layers are set to 2 and decline at a different rate when model layers increase. It should be noted that the training for a deeper model can become obviously tricky due to over-smoothing, a phenomenon of the node features tending to converge to the same vector. Compared with GGAN, however, the performance of Geom-GNN decreases sharper. Therefore, we make a conclusion that the introduction of geometric algebra may probably slow down the over-smoothing phenomenon of the GNN model.

Detailed performances with different hidden layer size

Different hidden layer dimensions would affect classification performance. The results are reported in Fig 3(b). We utilize a fixed 2-layer architecture and change the hidden layer dimension. The other hyperparameters are the same as the experiments investigating the influence of layer depth on model performance. As Fig 3 shows, There is no correlation between the increasing of hidden layer dimension with fixed network architecture and model performance. However, G3-GGAN and G4-GGAN show better performance compared with Geom-GNN, which implied that the model performance had been strengthened when introducing Geometric algebra.

Conclusion

This paper presents a novel neighborhood aggregation scheme, which develops geometric algebra into graph neural networks to learn graph embedding representation and reduce the parameter count in the model. We proposed a Graph Geometric Algebra Network(GGAN) consisting of G3-GGAN, and G4-GGAN to learn node and graph embeddings within Geometric Algebra space. By leveraging the properties of GA operation, our models achieve the most advanced performance on a range of well-known benchmark datasets for the two main tasks of node classification, and graph classification.

In future work, we will explore the extension of geometric algebra to dynamic graphs and time-series data for capturing temporal dynamics and evolving patterns. In addition, the application of geometric algebra in deeper graph convolutional networks will be further investigated to enhance representation learning and model capacity.

Discussion

The proposed Graph Geometric Algebra Network(GGAN) presents promising opportunities for advancing graph representation learning in various real-world applications. For instance, GGAN can enhance tasks in social network analysis, such as community detection and influence modeling, by capturing complex user interactions. In molecular and biomedical fields, its ability to handle high-dimensional and non-Euclidean structures makes it suitable for drug discovery, molecular property prediction, and biological network analysis. Additionally, GGAN's expressive embeddings can improve performance in knowledge graph tasks like link prediction and semantic search. Beyond these, GGAN has potential in cybersecurity for detecting fraud and cyber threats, as well as in optimizing infrastructure networks, including traffic prediction and logistics management.

However, the use of GGAN raises important ethical considerations, particularly in sensitive domains. For example, when applied to social networks, there is a risk of compromising user privacy through unintended information leakage in embeddings. Bias in training data may also be amplified, affecting fairness in applications like hiring or financial systems. Furthermore, the potential for misuse in surveillance or profiling underscores the need for responsible deployment with clear regulatory oversight. Addressing these concerns requires incorporating privacy-preserving techniques, fairness-aware training, and improving the interpretability of GGAN's predictions. By balancing its technical potential with ethical responsibility, GGAN can contribute meaningfully to both research and practice while minimizing risks.

Data availability
All data generated or analyzed during this study are included in this paper. The data of this work is available on request from the authors. The validated dataset are also available in the public repository, which can be found in https://github.com/daiquocnguyen/QGNN.

Received: 18 September 2024; Accepted: 24 December 2024
Published online: 02 January 2025

References
1. Hamilton, W. L., Ying, R. & Leskovec, J. Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 (2017).
2. Liu, K. et al. Chemi-net: a molecular graph convolutional network for accurate drug property prediction. *International journal of molecular sciences* **20**, 3389 (2019).
3. Wenzel, J., Matter, H. & Schmidt, F. Predictive multitask deep neural network models for adme-tox properties: learning from large data sets. *Journal of chemical information and modeling* **59**, 1253–1268 (2019).
4. Wang, Y. et al. User identity linkage across social networks via linked heterogeneous network embedding. *World Wide Web* **22**, 2611–2632 (2019).
5. CAI, T. *et al.* Target-aware holistic influence maximization in spatial social networks. *IEEE Transactions on Knowledge and Data Engineering* 1–1 (2020).
6. West, J. D., Wesley-Smith, I. & Bergstrom, C. T. A recommendation system based on hierarchical clustering of an article-level citation network. *IEEE Transactions on Big Data* **2**, 113–123 (2016).
7. Zhang, J. & Zhu, L. Citation recommendation using semantic representation of cited papers' relations and content. *Expert Systems with Applications* 115826 (2021).
8. Seo, Y., Loukas, A. & Perraudin, N. Discriminative structural graph classification. arXiv preprint arXiv:1905.13422 (2019).
9. Veličković, P. et al. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).
10. Pei, H., Wei, B., Chang, K. C.-C., Lei, Y. & Yang, B. Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020).
11. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035 (2017).
12. Maron, H., Ben-Hamu, H., Shamir, N. & Lipman, Y. Invariant and equivariant graph networks. arXiv preprint arXiv:1812.09902 (2018).
13. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018).
14. Liu, Q., Nickel, M. & Kiela, D. Hyperbolic graph neural networks. *Advances in neural information processing systems* **32** (2019).
15. Leman, A. & Weisfeiler, B. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya* **2**, 12–16 (1968).
16. Wang, R., Shen, M., Wang, X. & Cao, W. Rga-cnns: Convolutional neural networks based on reduced geometric algebra. *Sci. China Inf. Sci.* **64**, 1–3 (2021).
17. Su, H. & Bo, Z. Conformal geometric algebra based band selection and classification for hyperspectral imagery. In *2016 8th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 1–4 (IEEE, 2016).
18. Liu, Q. & Cao, W. Geometric algebra graph neural network for cross-domain few-shot classification. *Applied Intelligence.* **52**, 12422–12435 (2022).
19. Zhong, J. & Cao, W. Geometric algebra-based multiview interaction networks for 3d human motion prediction. *Pattern Recognition.* **138**, 109427 (2023).
20. Parcollet, T. et al. Quaternion recurrent neural networks. arXiv preprint arXiv:1806.04418 (2018).
21. Zhu, X., Xu, Y., Xu, H. & Chen, C. Quaternion convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 631–647 (2018).
22. Wu, J. et al. Deep octonion networks. *Neurocomputing.* **397**, 179–191 (2020).
23. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The graph neural network model. *IEEE transactions on neural networks* **20**, 61–80 (2008).
24. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).
25. Zheng, R., Chen, W. & Feng, G. Semi-supervised node classification via adaptive graph smoothing networks. *Pattern Recognition.* **124**, 108492 (2022).
26. Lin, X. et al. Exploratory adversarial attacks on graph neural networks for semi-supervised node classification. *Pattern Recognition* **133**, 109042 (2023).
27. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É. & Bouchard, G. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2071–2080 (PMLR, 2016).
28. Cao, Z., Xu, Q., Yang, Z., Cao, X. & Huang, Q. Dual quaternion knowledge graph embeddings. *In Proceedings of the AAAI Conference on Artificial Intelligence* **35**, 6894–6902 (2021).
29. Nguyen, D. Q., Nguyen, T. D. & Phung, D. Quaternion graph neural networks. arXiv preprint arXiv:2008.05089 (2020).
30. Dorst, L. & Mann, S. Geometric algebra: a computational framework for geometrical applications. *IEEE Computer Graphics and Applications* **22**, 24–31 (2002).
31. Clevert, D.-A., Unterthiner, T. & Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015).
32. Verma, S. & Zhang, Z.-L. Graph capsule convolutional neural networks. *ArXiv*[SPACE]arXiv: 1805.08090 (2018).
33. Zhang, M., Cui, Z., Neumann, M. & Chen, Y. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 4438–4445 (2018).
34. Maron, H., Ben-Hamu, H., Serviansky, H. & Lipman, Y. Provably powerful graph networks. *Advances in neural information processing systems* **32** (2019).
35. Xinyi, Z. & Chen, L. Capsule graph neural network. In *International conference on learning representations* (2019).

36. Chen, T., Bian, S. & Sun, Y. Are powerful graph neural nets necessary? a dissection on graph classification. arXiv preprint arXiv:1905.04579 (2019).
37. Xie, Y., Yao, C., Gong, M., Chen, C. & Qin, A. K. Graph convolutional networks with multi-level coarsening for graph classification. *Knowledge-Based Systems*. **194**, 105578 (2020).
38. Wang, Y., Wang, H., Jin, H., Huang, X. & Wang, X. Exploring graph capsual network for graph classification. *Information Sciences*. **581**, 932–950 (2021).
39. Nikolentzos, G., Dasoulas, G. & Vazirgiannis, M. Permute me softly: learning soft permutations for graph representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
40. Bouritsas, G., Frasca, F., Zafeiriou, S. & Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**, 657–668 (2022).
41. Bianchi, F. M., Grattarola, D., Livi, L. & Alippi, C. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence* **44**, 3496–3507 (2021).
42. Yu, B., Xu, X., Wen, C., Xie, Y. & Zhang, C. Hierarchical graph representation learning with structural attention for graph classification. In *Artificial Intelligence: Second CAAI International Conference, CICAI 2022, Beijing, China, August 27–28, 2022, Revised Selected Papers, Part II*, 473–484 (Springer, 2023).
43. Liu, W. et al. Locality preserving dense graph convolutional networks with graph context-aware node representations. *Neural Networks*. **143**, 108–120 (2021).
44. Wang, Z. et al. Location-aware convolutional neural networks for graph classification. *Neural Networks*. **155**, 74–83 (2022).
45. Abu-El-Haija, S., Kapoor, A., Perozzi, B. & Lee, J. N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *uncertainty in artificial intelligence*, 841–851 (PMLR, 2020).
46. Hu, Y. et al. Graph-mlp: Node classification without message passing in graph. arXiv preprint arXiv:2106.04051 (2021).
47. Yang, R., Dai, W., Li, C., Zou, J. & Xiong, H. Tackling over-smoothing in graph convolutional networks with em-based joint topology optimization and node classification. *IEEE Transactions on Signal and Information Processing over Networks* **9**, 123–139 (2023).
48. Yanardag, P. & Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1365–1374 (2015).
49. Yi, Y., Lu, X., Gao, S., Robles-Kelly, A. & Zhang, Y. Graph classification via discriminative edge feature learning. *Pattern Recognition*. **143**, 109799 (2023).
50. Han, X., Jiang, Z., Liu, N. & Hu, X. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, 8230–8248 (PMLR, 2022).
51. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
52. Sen, P. et al. Collective classification in network data. *AI magazine* **29**, 93–93 (2008).
53. Van der Maaten, L. & Hinton, G. Visualizing data using t-sne. *Journal of machine learning research*. **9** (2008).

Acknowledgements
This work was supported by National Natural Science Foundation of China under grant 61771322 and the Fundamental Research Foundation of Shenzhen under Grant JCYJ20220531100814033.

Author contributions
JQ.Zhong: Investigation, Formal analysis, Software, Methodology; WM.Cao: Supervision, Methodology, Funding acquisition.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to W.C.