Applied Intelligence (2023) 53:5697–5713
https://doi.org/10.1007/s10489-022-03780-1

# IV-GNN : interval valued data handling using graph neural network

Sucheta Dawn[1] · Sanghamitra Bandyopadhyay[1]

Accepted: 17 May 2022 / Published online: 1 July 2022

**Abstract**
Interval-valued data is an effective way to represent complex information where uncertainty, inaccuracy etc. are involved in the data space and they are worthy of taking into account. Interval analysis together with neural network has proven to work well on Euclidean data. However, in real-life scenarios, data follows a much more complex structure and is often represented as graphs, which is non-Euclidean in nature. Graph Neural Network is a powerful tool to handle graph like data with countable feature space. So, there is a research gap between the interval-valued data handling approaches and existing GNN model. No model in GNN literature can handle a graph with interval-valued features and, on the other hand, Multi Layer Perceptron (MLP) based on interval mathematics can not process the same due to non-Euclidean structure behind the graph. This article proposes an Interval-Valued Graph Neural Network, a novel GNN model where, for the first time, we relax the restriction of the feature space being countable without compromising the time complexity of the best performing GNN model in the literature. Our model is much more general than existing models as any countable set is always a subset of the universal set $\mathbb{R}^n$, which is uncountable. Here, to deal with interval-valued feature vectors, we propose a new aggregation scheme of intervals and show its expressive power to capture different interval structures. We validate our theoretical findings about our model for graph classification task by comparing its performance with those of the state-of-the-art models on several benchmark and synthetic network datasets.

**Keywords** Graph neural network · Interval-valued feature · Interval mathematics · Aggregation operator

## 1 Introduction

In today's Data driven era, data is treated as new oil to the digital economy. This data coming from several real life scenario follow high dimensional and complex structure. Also, they do not necessarily have a Euclidean structure behind them and can be represented better as Non-Euclidean data such as graphs and manifolds. For instance, in the e-commerce system [39], the interactions between users and products can be represented as graphs. Additionally, bio-active molecules and their bio-activity [12] can be modelled as graphical data. In citation networks [17], papers can be viewed as nodes of a graph, and the link between different papers via citation can be modelled as edges of that graph. One of the basic differences of this kind of Non-Euclidean spaced data and Euclidean data is, a straight line joining any two points in Non-Euclidean space is not necessarily the shortest path between them as that of in Euclidean space. Therefore, applying the Artificial Neural Network technique to solve different tasks on the Non-Euclidean domain is not that straight forward and has faced several challenges [34].

- Graph data is full of non-uniformity. Each graph consists of a different number of nodes, and each node in the graph has a variable number of neighbours. Hence, convolution type operations are not directly applicable here.
- Application of the existing Machine Learning algorithm is not always possible because instances are not independent here. In graphical data, the nodes share connections with other nodes via edges. So, to extract maximum information about the data while performing any specified task, it is necessary to capture this inter dependency among nodes.

To overcome these challenges, Graph Neural Network (GNN) has emerged as a powerful tool in last few years. The main idea behind GNN is to find low-dimensional vector

✉ Sucheta Dawn
suchetad_r@isical.ac.in

Sanghamitra Bandyopadhyay
sanghami@isical.ac.in

[1] Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India

embedding of nodes in a large graph, which will capture the structural information of the graph as well as the feature information of the nodes. This embedding can further be used to perform various types of prediction and analytical tasks on graphs using deep learning approaches. This idea enables us to use the computational capability of deep learning approaches in approximating most of the practically valuable functions on graphs. However, previous works on Graph Neural Network are focused on the graphs, where the input feature space is countable. Although, it is not uncommon for data to be recorded as intervals instead of precise point values in statistics. In general, interval-value data arises due to two types of situations. Firstly, when there is an uncertainty involved in the feature space, which needs to be captured and secondly, when there is a class, collection or group involved instead of individuals. The common example of interval valued data includes recording systolic and diastolic pressure, an individual's weekly expense ranger, daily temperature, stock price etc. Also, an important data types can be treated as interval-valued data. A potential approach to perform this transformation is to use visibility algorithm [18], i.e., assigning visibility range interval in both side (left and right) of a data point. The traditional approach to analyze the interval-valued data was to use the midpoint of the interval to a regression model [2]. Nevertheless, the Center Method (CM) does not consider the variations of the intervals. To overcome this drawback, Center-Range Method (CRM) uses two interval-valued regression models for mid-points and the ranges of the interval values [28]. However, in general, the mid-points and half ranges are related, which has not been taken into account by CRM [1]. Bivariate Center and Range Method (BCRM) uses two regression models using mid-points and half-ranges of the intervals, which take care of the effects of interval widths [3].

All these linear regression models can be used to analyze data with linear patterns. Also, there are several instances, where the inter connectivity between the entities is also important enough to capture. For example, infectious diseases such as the mumps in UK at country level in 2005 [See Fig. 1[1]] or the global pandemic COVID-19, the intensity of the disease in an area is highly dependent on the other areas linked with. The link might be in the form of people movement, weather patterns such as wind, geographical connection etc. Also, the daily new cases over a period of time can be modeled as an interval with minimum and maximum of new cases. Therefore, the number of infected in upcoming days or whether the transportation between two places should be stopped or not, these problems can be modeled as prediction task on graph with interval-valued feature. Our aim is to develop an appropriate GNN architecture, where the model accepts the interval valued feature, performs the embedding generation efficiently and finally, executes the specific task using those embeddings. Then an interesting question to be asked will be, why not use two end points of an interval as two separate features of a node and apply an existing GNN architecture accepting countable node feature of a graph. Answer to this question is although neural network has ability to capture relationship among different node features, interval is a quantitative measurement, where there is an order and the difference of two end points is meaningful. Our aim is to exploit this property of interval and develop an appropriate GNN architecture, where the model is capable enough to accept multiple intervals and output a single representation on its own. This will allow us to consider an interval as a unit through out the progress of the algorithm and perform the classification task as a function of the interval valued feature as well as the structure of the graph. Motivated by this situation, we introduce a new aggregation function of intervals, named as $agr_{new}$ and develop a neural architecture Interval-valued Graph Neural Network (IV-GNN) to apply on interval-valued data using MLPs and the newly developed $agr_{new}$ as its basic building blocks. Now, we give a summary of our main contributions here:

1. We introduce a new interval aggregation function $agr_{new}$ and precisely discuss its representational power.
2. We identify interval structures that were previously available interval aggregation schemes can not distinguish.
3. We develop a neural-based architecture Interval-Valued Graph Neural Network (IV-GNN), that can deal with interval-valued features.
4. We discuss the space and time complexities of our proposed algorithm and validate our theoretical findings through performing graph classification tasks on several datasets.

The rest of the paper is organized as follows. The related works on Graph Neural Network are reviewed in the Section 2. We have discussed about necessary basics of Graph Neural Network, Interval Mathematics and Aggregation Operator in the Section 3. The proposed framework for Interval-Valued Graph Neural Network (IV-GNN) is introduced in Section 5. The results of the experimental study are presented in Section 6, and finally, the concluding remarks are made in Section 7.

## 2 Related works

Encouraged by the commercial success of deep learning approaches in Euclidean structured data, a large number of methods have been introduced for Non-Euclidean data

[1]Figure source: https://www.newton.ac.uk/files/seminar/20140115093010001-153908.pdf

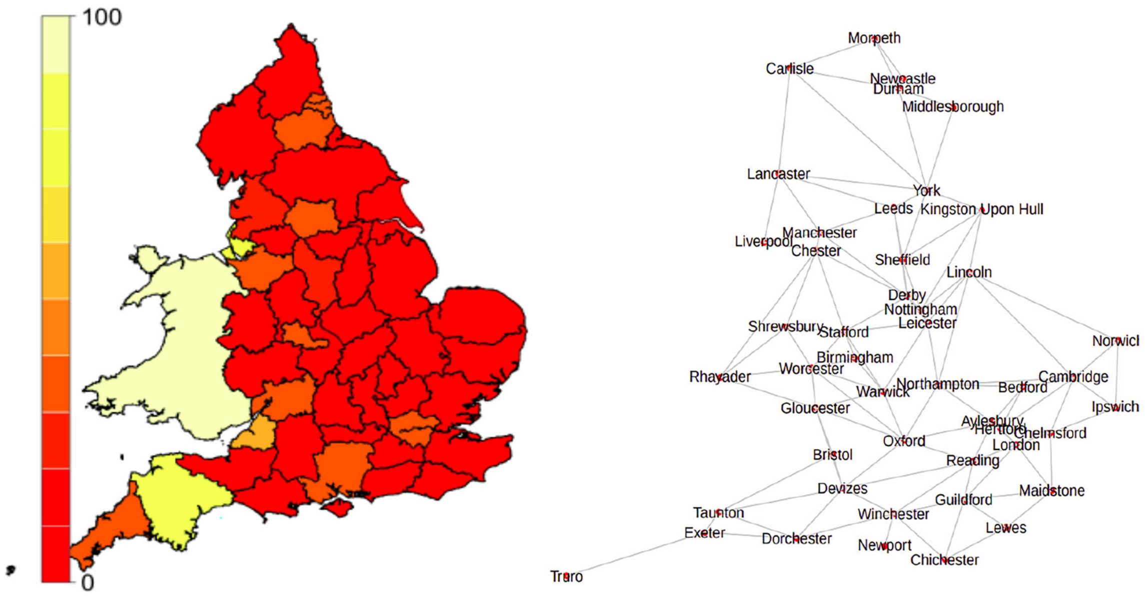

**Fig. 1** Left side: Weekly cases of Mumps at county level for 2005 in counties of England and Wales. Right side: Network structure using distances between county towns

as well [24]. These approaches primarily can be classified into two categories, namely spectral-based and spatial-based methods. The basic approach of spectral based GNN is to introduce filters from the graph signal processing perspective on a similar notion of the traditional convolutional neural network (CNN) [5]. However, due to high computational cost and lack of scalability of spectral-based approach, a relatively newer field of research, spatial-based GNN has gained popularity in recent years. These models can handle large graphs by aggregating feature information from the neighbouring nodes. We give an overview of a few models with spatial-based approach here,

1. One of the earliest works in this area, Graph Neural Network (GNN) [30] recursively updates latent node representations by exchanging information with the neighbouring nodes, until equilibrium is reached. The recurrent function is chosen to be a contraction mapping to ensure convergence.
2. Gated Graph Neural Networks (GGNN) [9] uses a gated recurrent unit as the recurrent function and use back-propagation through time (BPTT) for parameter learning. Hence, the condition on parameters to converge is no longer there, which reduces the number of steps.
3. Stochastic Steady-State Embedding (SSE) [10] uses a recurrent function that takes a weighted average of the states from previous steps and a new state to ensure the stability of the algorithm.
4. GraphSage [14] proposes a batch-training algorithm, which improves scalability for large graphs. It samples a fixed-sized neighbourhood of a node to aggregate information.
5. Graph Isomorphism Network GIN [35] is one of the best performing models reported in the literature. It has been claimed that both the GIN and WL test are equally powerful in a graph classification task. It imposes a constraint on the functions used in the model to be injective to achieve the maximum representational power of a GNN.
6. Higher order Graph Neural Network $k$-GNN [26] uses $k$-dimensional neighbourhood of a node to aggregate information from these and generate the low-dimensional representation for the mode.

**Limitations** All of these above discussed GNN-architectures have one limitation in common; that they can not process feature values, which are not from Euclidean space. Even the categorical features can also be handled by these models by converting them to integer data using integer encoding or one-hot encoding. But, when the feature value is continuous, to the best of our knowledge, no existing architecture in GNN literature can process it. However, there are several instances as mentioned earlier, where it is more convenient to represent the feature as interval and treat interval as a single unit throughout the progress of the algorithm. Therefore, in order to overcome this limitation of the existing GNN model, we aim to design a GNN model, which not only will accept node feature as intervals, also will outperform the existing GNN model on countable feature space. In this paper, a new interval aggregation scheme $agr_{new}$ has been introduced, satisfying all the necessary properties of a general aggregation function. Then, we have proposed Interval-Valued Graph Neural Network (IV-GNN) using $agr_{new}$ as aggregation operator, which can handle interval-valued features. We have performed graph classification tasks on four bioinformatics datasets, two social network datasets, and six synthetic datasets and have found that IV-GNN (with the proposed interval aggregation scheme) performs better than GNNs with various existing interval aggregation functions. Moreover, we demonstrate that our proposed model using degenerate interval-valued features (a special case of the more general IV-GNN model) outperforms other state-of-the-art approaches, that accept only countable features, for the graph classification task.

## 3 Background and definition

This section overviews the Graph Neural Network framework, basics of Interval Mathematics, and Aggregation Operators.

### 3.1 Graph neural network

In this sub-section, we formally present the notion of Graph Neural Networks and the related concepts.

Let $G = (V, E)$ be a graph, where $x_v$ is the node feature vector associated with a node $v \in V$. We want GNN to solve mainly two kinds of tasks.

– **Node Classification** Let every node $v \in V$ has label $y_v$ associated with it. Then the objective is to get a vector representation $z_v$ for $v$ in Euclidean space such that $y_v$ turns out to be a function of $z_v$, i.e. $y_v = f(z_v)$.
– **Graph Classification** Let $\{G_1, ..., G_N\}$ be a collection of graphs, where $\{y_1, ..., y_N\}$ is the set of associated labels of the graphs. Then our aim will be to learn a Euclidean representation $z_G$ for a particular graph $G$ such that $y_G$ turns out to be a function of $z_G$, i.e. $y_G = g(z_G)$.

Now we describe the forward propagation algorithm (Algorithm 1) of any GNN model, which is used to generate embedding of nodes using graph structure and node features.

**Algorithm 1** GNN framework for embedding generation.

```
Input: Graph $G(V, E)$ ; input features $\{x_v, \forall v \in V\}$ ; depth K ; aggregator functions $AGGREGATE^{(k)}$ ,∀k
    $\in \{1, 2, ..., K\}$ ; combine functions $COMBINE^{(k)}$ , ∀k $\in \{1, 2, ..., K\}$
Output: Vector representations $z_v$ for all $v \in$ V
$h_v^0 \leftarrow x_v, \forall v \in V$
for $k = 1, 2, ..., K$ do
    for $v \in V$ do
        $h_{N(v)}^k \leftarrow AGGREGATE^{(k)}(\{h_u^{k-1}; \forall u \in N(v)\})$
        $h_v^k \leftarrow COMBINE^{(k)}(h_v^{k-1}, h_{N(v)}^k)$
    end
end
$z_v \leftarrow h_v^K; \forall v \in V$
return $z_v$
```

In this algorithm, $N(v)$ stands for the neighbourhood of a vertex $v$. The main idea of GNN evolves around two functions *AGGREGATE*$^{(k)}$ and *COMBINE*$^{(k)}$. *AGGREGATE*$^{(k)}$ function is used to accumulate information from the neighbouring nodes, and $COMBINE^{(k)}$ function is used to update the existing representation vector of a node after $(k-1)$ iteration with the help of aggregated information from its neighbours. Here the iteration number stands for how many hop-neighbourhood of a node we want to consider to get its representation vector.

For graph classification, we have another function named READOUT, which will accept the representation of every node after the final iteration $K$ and predict the graph label.

$$z_G = READOUT(\{z_v | v \in V\})$$

Several AGGREGATE and COMBINE functions have been proposed in the GNN literature. In the GraphSAGE architecture [14], the AGGREGATE and COMBINE functions have been defined as follows,

$$h_v^k = \sigma(W_2.CONCAT(h_v^{k-1}, MAX(\{ReLU(W_1.h_u^{k-1}), \forall u \in N(v)\}))) \quad (1)$$

where $W_1$ and $W_2$ are learnable weight matrices, CONCAT represents vector concatenation. $\sigma$ is a non linear function.

In Graph Convolution Network [17], the AGGREGATE and COMBINE function can be defined as,

$$h_v^k = ReLU(W.MEAN\{h_u^{k-1}, \forall u \in N(v) \cup \{v\}\}) \quad (2)$$

In Graph Isomorphism Network [35], the model uses a sum aggregator over max or mean aggregator due to its more discriminative power.

GIN updates existing node representations as,

$$h_v^k = MLP^k((1+\epsilon^k).h_v^{k-1} + \Sigma_{u\in N(v)} h_u^{k-1}) \tag{3}$$

MLP represents a multi-layer perceptron, and $\epsilon$ can be a parameter that needs to be learned or assigned as a fixed scalar. Any spatial-based GNN can be at most as powerful as WL test [35]. This network is one of those GNN models, which is equally powerful as the WL test distinguishing a broad class of graphs effectively and efficiently.

### 3.2 Interval mathematics

The works discussed so far take care of the situation when the feature space is countable. Before generalizing this idea to interval-valued feature space, we introduce the interval mathematics and properties of any aggregating function. Note that, if the node feature is from countable space, it can be considered a degenerate interval, i.e. the start and end points of the interval are same.

Let us consider $\mathcal{U} = \{[a,b] | 0 \le a \le b \le 1\}$ along with $\cap_0$ and $\cup_0$ defined by

$$[x_1, x_2] \cap_0 [y_1, y_2] = [min(x_1, y_1), min(x_2, y_2)]$$

$$[x_1, x_2] \cup_0 [y_1, y_2] = [max(x_1, y_1), max(x_2, y_2)]$$

$(\mathcal{U}, \cap_0, \cup_0)$ forms a complete lattice. A partially ordered set is said to be a complete lattice if all subsets have both a supremum (join) and an infimum (meet). We denote aggregation operator as $agr_0$ which is based on $\cap_0$.

Now to assign the value of the aggregation so that it takes care of every individual's opinions, the aggregated interval needs to be defined as an interval that lies in the intersection of everyone's opinion. If we define the aggregator function as $\cap_0$ as defined above, then the function is biased towards the interval with a lower value. Therefore, to overcome this drawback, in [13], $(\mathcal{U}, \cap_e, \cup_e)$ is defined as a lattice where the greatest lower bound, $\cap_e$ and the least upper bound, $\cup_e$ are defined as follows,

$$\begin{aligned}[x_1, x_2] \cap_e [y_1, y_2] &= [max(x_1, y_1), min(x_2, y_2)], \\ &\quad \text{if } max(x_1, y_1) \le min(x_2, y_2) \\ &= [min(x_2, y_2), min(x_2, y_2)] \text{ , otherwise}\end{aligned}$$

$$\begin{aligned}[x_1, x_2] \cup_e [y_1, y_2] &= [max(x_2, y_2), max(x_2, y_2)], \\ &\quad \text{if} x_1 = x_2, y_1 = y_2 \\ &= [max(x_1, y_1), max(x_2, y_2)], \\ &\quad \text{if } x_1 = x_2 < y_1 < y_2 \\ &= [min(x_1, y_1), max(x_2, y_2)] \text{ , otherwise}\end{aligned}$$

$(\mathcal{U}, \cap_e, \cup_e)$ is a complete lattice. We denote aggregation operator as $agr_e$ which is based on $\cap_e$.

### 3.3 Aggregation operators

Any aggregation operator [6] for fixed $n \ge 2$ is defined by a function $agr : [[0,1]\times[0,1]]^n \rightarrow [0,1]\times[0,1]$ fulfilling at least the two following axioms.

- **Boundary conditions** $agr(I_{min}, I_{min}, \dots, I_{min}) = I_{min}$, $agr(I_{max}, I_{max}, \dots, I_{max}) = I_{max}$, where $I_{min}$ and $I_{max}$ are minimal and maximal possible inputs respectively.
- **Monotonic increasing** $\forall (I_1, I_2, \dots, I_n), (J_1, J_2, \dots, J_n) \in [[0,1]\times[0,1]]^n$ such that $I_i \le J_i, \forall i \in \mathbb{N}$ then $agr(I_1, I_2, \dots, I_n) \le agr(J_1, J_2, \dots, J_n)$

Besides these properties, we want our aggregations to satisfy two additional axioms.

- **Symmetry** $agr(I_1, I_2, \dots, I_n) = agr(I_{p(1)}, I_{p(2)}, \dots, I_{p(n)})$ for any permutation $p$ on $\mathbb{N}^n$
- **Idempotency** $agr(I, I, \dots, I) = I, \forall I \in [0,1]\times[0,1]$

## 4 Theoretical framework

We introduce a new order relation $\subseteq_{new}$ on $\mathcal{U}$ such that $(\mathcal{U}, \subseteq_{new}, \cap_{new}, \cup_{new})$ forms a lattice and will give a comparable study against two previously discussed order relation.

### 4.1 Definition

$\subseteq_{new}$ is a binary relation on $\mathcal{U}$ defined as below

$$\begin{aligned}[x_1, x_2] \subseteq_{new} [y_1, y_2] &\text{ if } (y_1 < x_1) \text{ or} \\ &(x_1 = y_1 \text{ and } x_2 \le y_2).\end{aligned}$$

### 4.2 Proposition

$(\mathcal{U}, \subseteq_{new})$ forms a poset. To show $(\mathcal{U}, \subseteq_{new})$ forms a poset, we have to show the relation $\subseteq_{new}$ is reflexive, antisymmetric, and transitive.

- Reflexivity

  $[x_1, x_2] \subseteq_{new} [x_1, x_2]$ as $x_1 = x_1$ and $x_2 \le x_2$

- Antisymmetricity

  $$\begin{aligned}[x_1, x_2] \subseteq_{new} [y_1, y_2] \Longrightarrow\ & y_1 < x_1 \text{or} \\ & x_1 = y_1 \text{ and } x_2 \le y_2 \dots .(i)\end{aligned}$$

  $$\begin{aligned}[y_1, y_2] \subseteq_{new} [x_1, x_2] \Longrightarrow\ & x_1 < y_1 \text{ or} \\ & y_1 = x_1 \text{ and } y_2 \le x_2 \dots .(ii)\end{aligned}$$

  (i) and (ii) imply

  $x_1 = y_1$ and $x_2 = y_2$.

Hence ,

$[x_1, x_2] = [y_1, y_2]$

– Transitivity can be shown similarly.

## 4.3 Proposition

$(\mathcal{U}, \subseteq_{new})$ forms a lattice. $(\mathcal{U}, \subseteq_{new})$ is a lattice where the greatest lower bound, say $\cap_{new}$ and the least upper bound, say $\cup_{new}$ are defined as follows,

$$\begin{aligned}[x_1, x_2] \cap_{new} [y_1, y_2] &= [\max(x_1, y_1), \min(x_2, y_2)] \\ &\quad if\ \max(x_1, y_1) \leq \min(x_2, y_2) \\ &\quad \leq \max(x_2, y_2) \neq 1 \\ &= [\max(x_1, y_1), 1], \text{otherwise}\end{aligned}$$

$$\begin{aligned}[x_1, x_2] \cup_{new} [y_1, y_2] &= [\min(x_1, y_1), \max(x_2, y_2)] \\ &\quad \text{if} \max(x_2, y_2) \neq 1 \\ &= [\min(x_1, y_2), \min(x_2, y_2)], \\ &\quad \text{otherwise}\end{aligned}$$

## 4.4 Proposition

$(\mathcal{U}, \subseteq_{new})$ forms a bounded lattice. For an arbitrary interval $[x, y] \in \mathcal{U}$, we have $0 \leq x \leq y \leq 1$. Hence $[1, 1] \subseteq_{new} [x, y] \subseteq_{new} [0, 1]$. We denote aggregation operator as $agr_{new}$, which is based on $\cap_{new}$.

## 4.5 Proposition

$agr_{new}$ satisfies all four conditions of aggregation function. $agr_{new}$ satisfies boundary conditions where [1, 1] and [0, 1] are the minimal and maximal possible inputs respectively.

## 5 Interval-valued graph neural network (IV-GNN)

We develop a general GNN model where the feature space need not be countable. Releasing this constraint on the feature space, our model can capture the graph's structural properties and can extract useful information from interval-valued features of the nodes. As a result, our proposed architecture is much more general in nature and to the best of our knowledge, no existing models of GNN in literature can accept nodes' feature, which are intervals.

### 5.1 AGGREGATE and UPDATE function of IV-GNN

As we have already discussed that spatial-based GNN architecture has two primary functions, namely *AGGREGATE* and *COMBINE*, we use our newly develop interval aggregation operator $agr_{new}$ function to aggregate the neighbouring nodes' embedding to extract maximum information out of it. As shown in the Fig. 2, we develop our model with aggregation and update function on $k$-th iteration defined as

$$h_v^k = \Phi(h_v^{k-1}, agr_{new}(\{h_u^{k-1} : u \in N(v)\})) \tag{4}$$

, where $\Phi$ is the update function. To choose $\Phi$, we take the help of the Universal Approximation Theorem [15], which states that, using the multi-layer feed-forward architecture in a neural network framework, makes it a universal approximator of any continuous functions under mild assumptions on the activation function [23]. However, the limitation of continuity of the function was released much later. It has been shown that, a single hidden layer feed-forward neural network (SLFNNs) can approximate any real, piece wise continuous function almost uniformly [25]. Therefore, IV-GNN has the updating step as,

$$\begin{aligned}h_v^k = MLP^k(&agr_{new}((1+\epsilon^k)h_v^{k-1}, \\ &agr_{new}\{h_u^{k-1} : u \in N(v)\}))\end{aligned} \tag{5}$$

### 5.2 Details of updation step

The proposed Interval-Valued Graph Neural Network (IV-GNN) operates on interval- valued input and output in order to generate embeddings. As discussed earlier, for any given node, the newly developed aggregation function $agr_{new}$ has been used twice in every updation step. Firstly, it will accumulate the neighbours' intervals and express them as a single interval. And, secondly, it will combine neighbourhood information with the node's itself. We have discussed the $agr_{new}$ in detail in the previous section. Now, we give the details of the neural architecture of our model.

#### 5.2.1 Neural architecture of IV-GNN

One of the basic building blocks of our proposed model IV-GNN is Multi Layer Perceptron (MLP). However, unlike the commonly used architecture, the MLP used in the proposed model deals with interval-valued inputs and outputs, but the weights and biased are single-valued [29]. In order to understand, we assume that, the MLP has only one hidden layer with $h$ units.

Let us consider the interval-valued inputs as $X_i = [X_i^{lower}, X_i^{upper}]$, with $i = 1, 2, ..., n$. The output of the $j$-th hidden unit is the single-value weighted linear combination of inputs and the bias as follows,

$$\begin{aligned}S_j =< S_j^1, S_j^2 >=< w_{j0} + \sum_{i=1}^{n} w_{ji} \frac{X_i^{lower} + X_i^{upper}}{2}, \\ \sum_{i=1}^{n} |w_{ji}| \frac{X_i^{upper} - X_i^{lower}}{2} >\end{aligned}$$

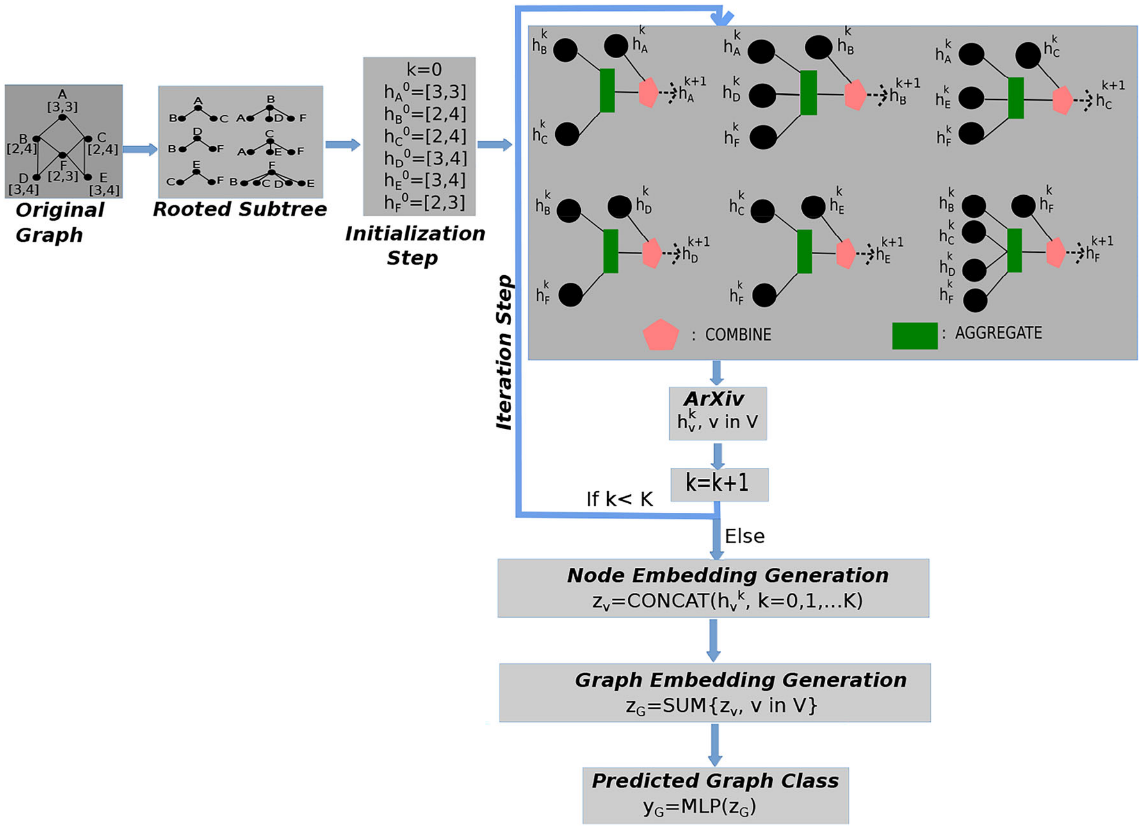

**Fig. 2** An overview of the proposed framework: IV-GNN

After that, *tanh* function has been used as activation function,

$$A_j = tanh(S_j) = [tanh(S_j^1 - S_j^2), tanh(S_j^1 + S_j^2)]$$

Finally, the output is the linear combination of the hidden layer output and bias, which is the output of the whole updation step.

## 5.3 Graph-level readout function of IV-GNN

The purpose of this graph-level read-out function is to get embedding of the graph using the embedding of the nodes. If we want to perform jobs like node classification [22] or link prediction [8] within a graph then the node embedding using aggregation and update function at node level are sufficient.

While selecting the Graph-level READOUT function, we want to focus on the following aspects.

– We want to use the structural information/node embedding that we have got after every iteration. It may so happen that the node embedding from an earlier iteration captures more information about the graph rather than the final iteration.
– The graph-level function should be injective to have our GNN variant as powerful as WL-test of isomorphism by distinguishing between different structures/node features.

Hence, to achieve a skip connection like architecture similar to Jumping Knowledge [36] and maximal distinguishing power, we concatenate the SUM of the node embeddings after every iteration.

$$z_G = CONCAT(SUM(\{h_v^k | v \in G\}) | k = 0, 1, \ldots, K) \quad (6)$$

## 5.4 Challenging structures for $agr_0$ and $agr_e$

The main idea of choosing a powerful aggregator is to capture and compress the amount of structural and feature information from the nodes in its aggregated output value. Also, the aggregator function should be permutation invariant. That is, the order of the interval during aggregation should be immaterial. In this context, $agr_0$ , $agr_e$ and $agr_{new}$, all are satisfying this condition. In Fig. 3, we have shown the ranking of three aggregation functions pictorially with respect to their representational power. We have denoted the root node as the red node and the adjacent nodes of the root node as the black node whose features need to be aggregated and combined with the root node. In Table 1, we have illustrated these facts with the help of examples.

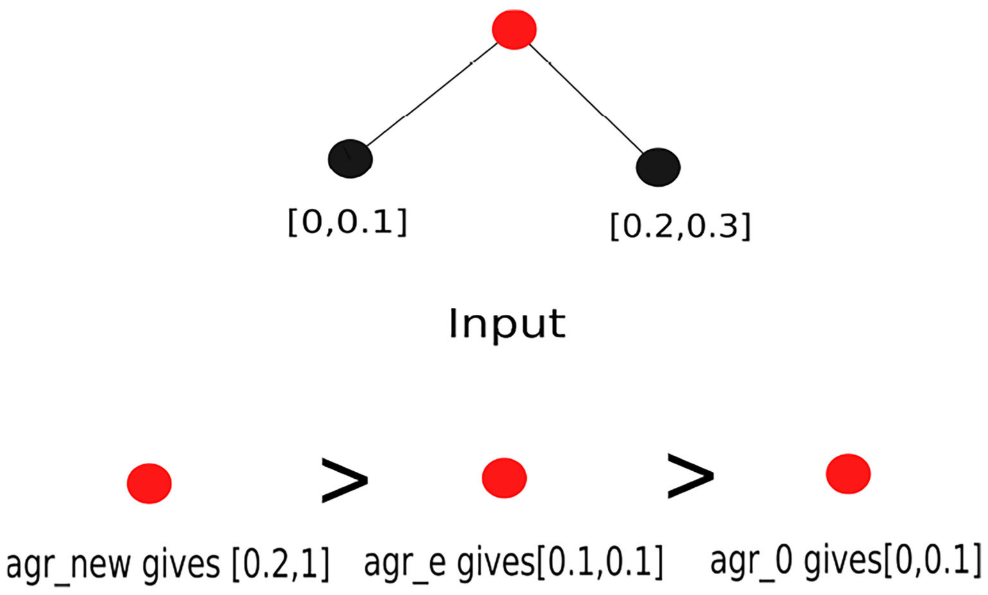

**Fig. 3** Ranking by expressive power for $agr_0$, $agr_e$, $agr_{new}$. Among these three aggregators, $agr_{new}$ has the maximum ability to capture the structural and feature related information. One thing to notice that, the right end point of an resulting interval equals to 1expresses the fact that two aggregating intervals are non overlapping. $agr_e$ is equally powerful as $agr_{new}$ when two intervals have non-nullintersection. $agr_0$ captures the smaller interval (according to the order relation) and ignores the other interval

In the Fig. 4a, we have three intervals, $I_1 = [0.1, 0.2]$, $I_2 = [0.1, 0.3]$ and $I_3 = [0.15, 0.3]$. We construct two sets of intervals $S_1$ and $S_2$, where $S_1 = \{I_1, I_2\}$ and $S_2 = \{I_1, I_3\}$, which need to be aggregated. Now,

$$agr_0(S_1) = agr_0(S_2) = I_1$$

Hence, in this case $agr_0$ fails to capture the desired information about the intervals. However, $agr_e$ and $agr_{new}$ will give the aggregated intervals as the intersecting sub intervals.

$$agr_e(S_1) = agr_{new}(S_1) = [0.1, 0.2]$$

$$agr_e(S_2) = agr_{new}(S_2) = [0.15, 0.2]$$

In Fig. 4b, two sets of intervals need to be aggregated $S_3 = \{I_1, I_4\}$ and $S_4 = \{I_1, I_5\}$, where $I_4 = [0.3, 0.4]\}$ and $I_5 = [0.2, 0.3]$. Here also, $agr_0$ will fail to distinguish between them.

$$agr_0(S_1) = agr_0(S_2) = I_1$$

But according to definition, $agr_e$ will give same degenerate interval [0.2, 0.2] for both $S_3$ and $S_4$. However $agr_{new}$ can differentiate between them as it will aggregate and give the resultant intervals as [0.3, 1] and [0.2, 0.2] for $S_3$ and $S_4$ respectively.

$$agr_e(S_3) = agr_e(S_4) = [0.2, 0.2]$$

$$agr_{new}(S_3) = [0.3, 1],$$
$$agr_{new}(S_4) = [0.2, 0.2]$$

As $I_1$ and $I_4$ are non-intersecting, $agr_{new}$ will be able to capture the uncertainty and express it by assigning a broader interval.

The $agr_e$ will perform well if the two intervals have a non-null intersection. So, whenever the node features are not very diverse, $agr_e$ will be as powerful as the $agr_{new}$ aggregator.

## 5.5 Model training

To estimate model parameters of IV-GNN, we need to specify an objective function to optimize. Since the task we focus on in this work is Graph Classification task, loss is computed as the sum of the difference between the actual graph class and the predicted graph class for the graphs in the dataset.

## 5.6 Memory and space complexity of training the embedding generation process of IV-GNN

In order to find the worst case scenario, we assume that all $||E||$ edges are connected to all $||V||$ nodes of the graph $G$. As IV-GNN is a full-batch gradient descent process, it requires storing all the embeddings found from the intermediate layers, which requires $O(Kn)$ storage for one node. Here, $K$ denotes the number of layers and $n$ denotes the dimension of the embedding space. For the sake of simplicity, we keep the embedding space dimension same for every layer. Furthermore, at every layer, a weight matrix of size $n \times n$ is involved, which includes $O(Kn^2)$

**Table 1** Comparison between three interval aggregator

| Interval 1 | Interval 2 | $agr_0$ | $agr_e$ | $agr_{new}$ |
|---|---|---|---|---|
| [0.1, 0.2] | [0.1, 0.3] | [0.1, 0.2] | [0.1, 0.2] | [0.1, 0.2] |
| [0.1, 0.2] | [0.15, 0.3] | [0.1, 0.2] | [0.15, 0.2] | [0.15, 0.2] |
| [0.1, 0.2] | [0.3, 0.4] | [0.1, 0.2] | [0.2, 0.2] | [0.2, 1] |
| [0.1, 0.2] | [0.2, 0.3] | [0.1, 0.2] | [0.2, 0.2] | [0.2, 0.2] |

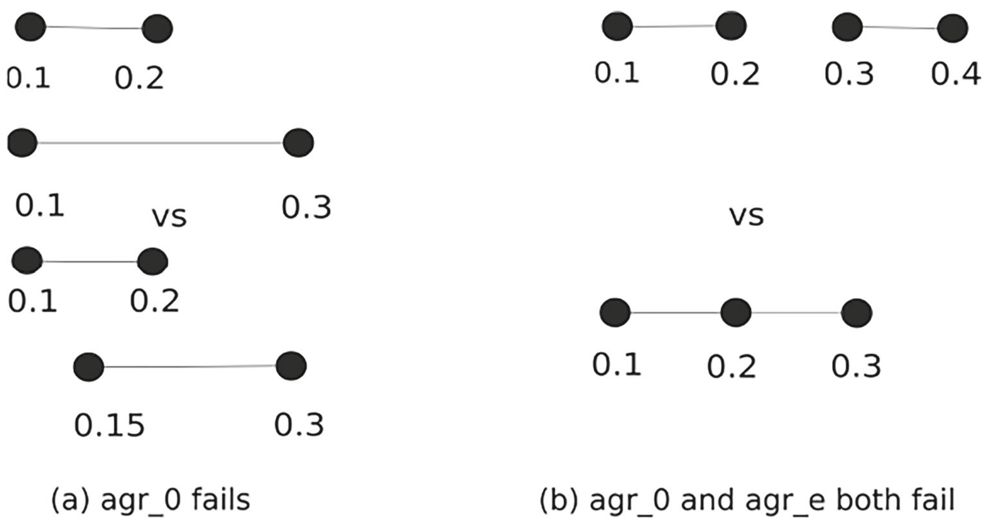

**Fig. 4** Examples of interval structures that $agr_0$ and $agr_e$ fail to distinguish. In the left picture, $agr_0$ is giving same aggregated interval even though two sets of intervals are different. In the right picture, $agr_0$ and $agr_e$, both are unable to recognise the differences between the two sets of intervals

storage in total. Therefore, overall IV-GNN has a space-complexity of $O(K||V||n + Kn^2)$. Now, we illustrate the time-complexity of our proposed model. As discussed previously, IV-GNN stores intermediate embeddings of every node, generated from each lower layer. In contrary to mini-batch algorithm like GraphSAGE [14], IV-GNN utilizes those saved embeddings and reuses those in the upper layer. Therefore, at every layer, previously layers' embeddings are multiplied with the weight matrix of size $n \times n$, which includes $n^2$-many multiplications, followed by some element-wise operations. Therefore, as a whole, for $K$ many layers and $||V||$ many nodes, IV-GNN has time complexity $O(K||V||n^2 + K||E||n)$.

## 6 Experiments

This section discusses the dataset used for experiments, and we evaluate our theoretical findings by comparing the training and test set performances of IV-GNN on the synthetic and real-life datasets.

### 6.1 Datasets

We have used six synthetic datasets, four bioinformatics datasets and two social network datasets to demonstrate the efficiency of our model.

1. Synthetic Datasets: Our basic idea is to generate random graphs with a various number of nodes. Then based on two topological properties, we give tag and feature interval to the nodes and classify every graph in the datasets. The summary statistics of these synthetic datasets are provided in Table 2. The topological properties are listed below,

   – **Density** The density of a graph is defined as the ratio of the number of edges and the number of nodes [19], i.e.,

   $$\text{density}(G) = \frac{|E|}{|V|}$$

   where $G = (V, E)$ denotes a graph, $V$ is the set of nodes and $E$ is the edge set of the graph $G$. Average density of the dataset $D = \{G_i, i = 1, ..., n\}$ can be calculated as,

   $avg\_density = \frac{\sum_{i=1}^{n} density(G_i)}{n}$
   We assign,

$$\text{Graph class (G)} = \begin{cases} 1, & \text{if density } (G) < \text{avg\_density.} \\ 0, & \text{otherwise.} \end{cases}$$

   We assign,

   $$\text{tag}(v) = \begin{cases} 1, & \text{if degree } (v) < \text{average degree.} \\ 0, & \text{otherwise.} \end{cases}$$

   where average degree is calculated over all nodes in the dataset (Fig. 5).

**Table 2** Statistics of the synthetic datasets

| Dataset | Size | Classes | Avg. nodes | labels |
|---|---|---|---|---|
| SYNTHETIC_1_200 | 200 | 2 | 19.94 | 2 |
| SYNTHETIC_1_1000 | 1000 | 2 | 19.83 | 2 |
| SYNTHETIC_1_2000 | 2000 | 2 | 19.92 | 2 |
| SYNTHETIC_2_200 | 200 | 2 | 19.94 | 23 |
| SYNTHETIC_2_1000 | 1000 | 2 | 19.83 | 25 |
| SYNTHETIC_2_2000 | 2000 | 2 | 19.92 | 25 |

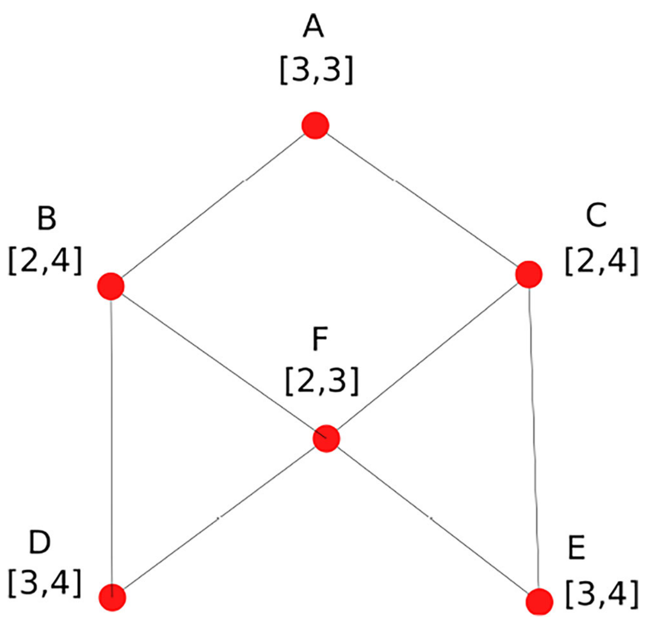

**Fig. 5** Explanation of assigning feature interval to a node. For the node A, it has two neighbours of degree 3, resulting $d_{min} = d_{max} = 3$. Hence the feature interval$(A) = [d_{min}, d_{max}] = [3, 3]$. Similarly, node F has 4 neighbours of degree 2, 2, 3, 3. Hence $d_{min}$ and $d_{max}$ are 2 and 3 respectively. Therefore, feature interval$(F) = [2, 3]$

To assign the interval valued feature to a node $v$, we follow the following rule,

$$\text{feature interval}(v) = \begin{cases} [d_{min}, d_{max}], & \text{if} |N(v)| > 1 \\ [-1, d_{max}], & \text{if} |N(v)| = 1 \\ [-1, 0], & \text{otherwise.} \end{cases}$$

where

$d_{min} = \min_{u \in N(v)} degree(u)$,

$d_{max} = \max_{u \in N(v)} degree(u)$

We have created three datasets, SYNTHETIC_1_200, SYNTHETIC_1_1000, SYNTHETIC_1_2000, with 200, 1000, 2000 graphs, respectively, according to this construction.

– **Average clustering coefficient** The clustering coefficient $c(u)$ of a node $u$, is a measure of the likelihood that any two neighbors of $u$ are connected [21]. Mathematically, the clustering coefficient of a node $u$ can be formulated as:

$$c(u) = \frac{\lambda(u)}{\tau(u)}.$$

where $\lambda(u)$ is the number of triangles engaging the node $u$. By triangle, we mean complete graph with three nodes and

$$\tau(u) = \frac{degree(u)(degree(u) - 1)}{2},$$

i.e., the number of triples a node $u$ has.

In other words, the clustering coefficient for node $u$ is the ratio of the actual number of edges between two nodes from the neighbours of $u$ and the maximally possible numbers of edges between them. The clustering coefficient $C(G)$ of a graph $G$ is the average of $c(u)$ taken over all the nodes in the graph, i.e.,

$$C(G) = \frac{1}{n} \sum_{i=1}^{n} c(u_i)$$

Average clustering coefficient of the dataset $D = \{G_i, i = 1, ..., n\}$ can be calculated as,

$avg\_cluster = \frac{\sum_{i=1}^{n} C(G_i)}{n}$

We assign,

$$\text{Graph class}(G) = \begin{cases} 1, & \text{if } C(G) < avg\_cluster \\ 0, & \text{otherwise.} \end{cases}$$

We use node's degree as its tag.

$$\text{tag}(v) = degree(v)$$

To assign the interval valued feature to a node $v$, we follow the following rule,

$$\text{feature interval}(v) = [c_{min}, c_{max}]$$

where

$c_{min} = \min_{u \in N(v)} c(u)$,

$c_{max} = \max_{u \in N(v)} c(u)$

We have created three datasets SYNTHETIC_2_200, SYNTHETIC_2_1000, SYNTHETIC_2_2000 with 200, 1000, 2000 graphs respectively according to this construction.

2. Bio-informatics datasets: 4 datasets MUTAG, PROTEINS, PTC and NCI1 [37] have been used for our experiment. The summary statistics of these bioinformatic datasets are provided in Table 3.

   – MUTAG consists of 188 graphs which have 7 discrete node labels. Each graph in the dataset represents a chemical compound [11].
   – In the dataset PROTEINS, nodes represent secondary structure elements (SSEs) and two nodes share an edge if they appear as adjacent in the amino-acid sequence. Graph nodes have 3 different labels such as helix, sheet or turn [4].
   – PTC includes 344 chemical compounds that describes the carciogenicity for male and female rats having 19 discrete node labels [32].
   – NCI1 is a balanced dataset with 4100 nodes with 37 discrete labels, published by the National Cancer Institute (NCI). It contains chemical compounds,

**Table 3** Statistics of the Bioinformatics datasets used

| Dataset | Size | Classes | Avg. nodes | labels |
|---|---|---|---|---|
| MUTAG | 188 | 2 | 17.9 | 7 |
| PTC | 344 | 2 | 25.5 | 19 |
| PROTEINS | 1113 | 2 | 39.1 | 3 |
| NCI1 | 4110 | 2 | 29.8 | 37 |

that are found to have the ability to suppress or inhibit the growth of a panel of human tumor cell lines [33].

3. Social network datasets: 2 datasets, IMDB-BINARY and COLLAB [37] have been used for our experiment. The summary statistics of these datasets are provided in Table 4.
   - Movie Collaboration Dataset: IMDB-BINARY is a dataset of movie collaboration, wherein in each graph, nodes represent actors/actresses and two nodes share an edge if two actors/actresses act in the same movie. There are two graph classes Action and Romance genres.
   - Scientific collaboration dataset: COLLAB is a dataset of scientific collaboration, acquired from 3 public collaboration datasets, namely High Energy Physics, Condensed Matter Physics and Astro physics [20]. In [31], ego-networks of various researchers from each field has been generated, and each graph has a label according to the field of the researcher. Now the task will be to determine an ego-collaboration graph's label of a researcher.

## 6.2 Performance of IV-GNN

Our goal is to allow the model to capture structural information and feature information from the graph. Therefore, we like to evaluate our model IV-GNN on Graph Classification task with interval-valued features of the nodes.

### 6.2.1 Data preparation

To convert the feature values into intervals, we prepare the data as follows: rather than using the tag of a node as a node feature, we give a bias of $k_1$ and $k_2$, where $k_1$ and $k_2$ are selected from the normal distribution $N(0, 1)$ to generate an interval-valued feature for that particular node. For example, if in a graph, a node has a tag as c, then we assign $[c - k_1, c + k_2]$ as its feature interval.

**Table 4** Statistics of the Social network datasets used

| Dataset | Size | Classes | Avg. nodes | labels |
|---|---|---|---|---|
| IMDBBINARY | 1000 | 2 | 19.8 | - |
| COLLAB | 5000 | 3 | 74.5 | - |

### 6.2.2 Baselines

We evaluate our IV-GNN by comparing it with other frameworks with different interval aggregation operators. As there is no existing model, who can handle interval-valued features of the nodes, we use existing interval aggregation operators with same neural architecture as of IV-GNN, which will show expressive power of $agr_{new}$ against that of others experimentally. The details of the baselines are discussed below.

- $agr_0$-based GNN In this model, we choose $agr_0$ for interval-aggregation and $SUM$ as Graph-Level *READOUT* function.
- $agr_e$-based GNN In this model, interval-aggregation operator $agr_e$ has been used as *AGGREGATE* function. Like before, we use *SUM* as the Graph-Level *READOUT* function.

We report the training set performance and test set performance in the Fig. 6 and Table 5 respectively.

## 6.3 Performance with degenerate interval: A special case of IV-GNN

In order to examine the performance of IV-GNN for countable features, we compare IV-GNN with respect to the state-of-the-art approaches which accept countable feature-value. For this experiment, we treat the exact value of the feature as a degenerate interval, i.e., we use exact feature value as same starting and end points of the interval. We report the training set performance and test set performance in the Fig. 7 and Table 6 respectively.

### 6.3.1 Baselines

We compare IV-GNN with *three*, state-of-the-art GNN models, as briefly discussed below.

- GraphSage [14]: This uses a sampled neighbourhood and aggregates information from these to generate the Euclidean representation of the nodes.
- GCN [17]: This model uses the MEAN-pool of the information from the neighbouring nodes and then combines the changes with the existing feature of a node using a nonlinear function.

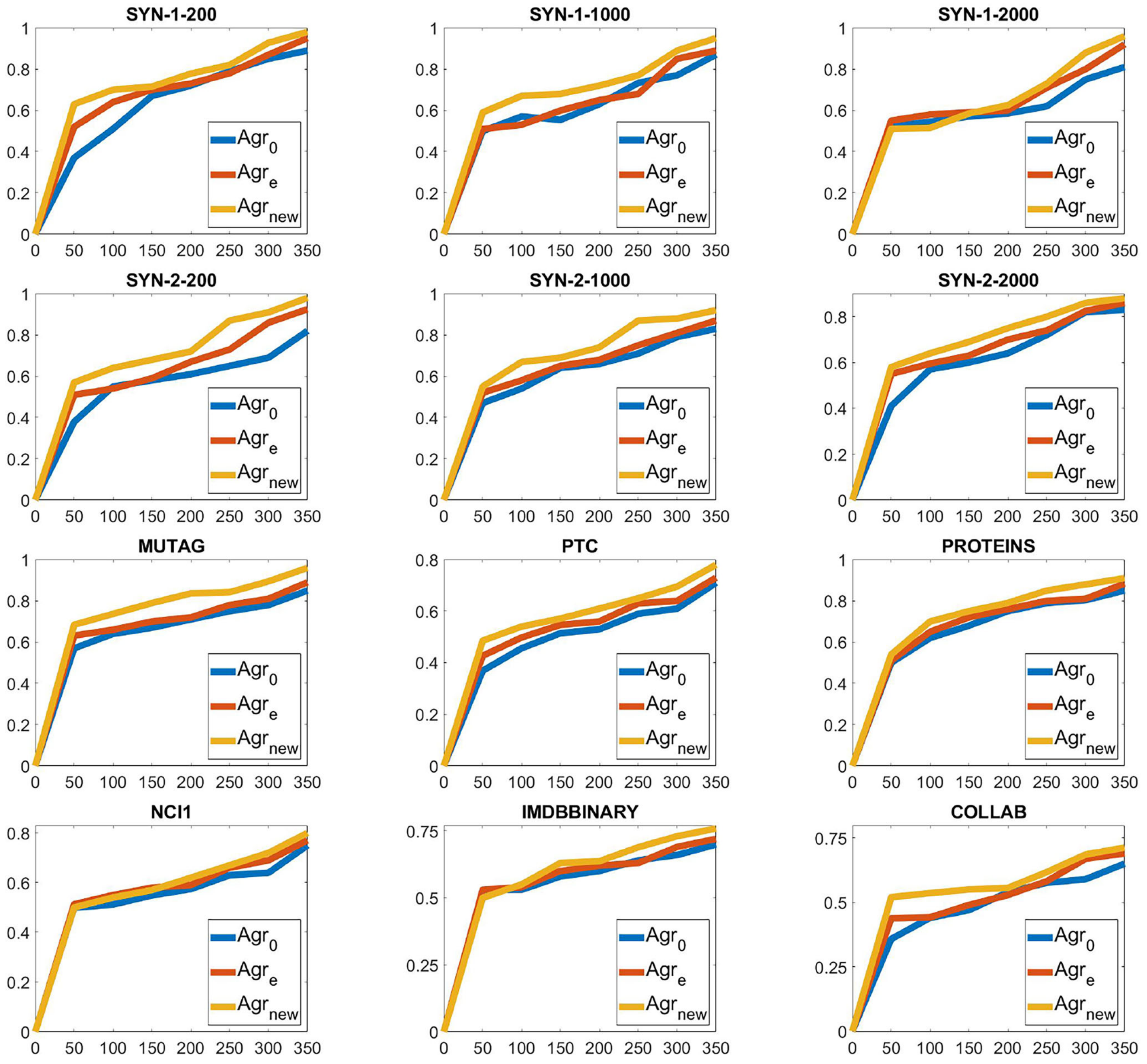

**Fig. 6** Training set performance of IV-GNN and less powerful interval valued feature accepting GNNs. X-axis and Y-axis show the epoch number and the accuracy of training respectively

- GIN [35]: This model uses SUM as its aggregation function because of its maximal discriminative power and a nonlinear function as a combine function.

## 6.4 Parameter settings

For IV-GNN, we adopt the parameter settings based on our hyperparameter study [see Section 6.7], and details are as follows

- We have used 5-layers of each GNN block where every MLP will have 2-layers excluding the input layer.
- Each hidden layer has {32, 128} hidden units. We have used Batch Normalization in each hidden layer.
- We have used Adam optimizer [16] with the initial learning rate 0.01 and decay the learning rate by 0.5 after every 50 epochs.
- Input batch size of training is {16, 64}.
- The final layer dropout is 0.5 [38].

**Table 5** Test set classification accuracies in percentage

| Dataset | $agr_0$-based model | $agr_e$-based model | $agr_{new}$-based model IV-GNN |
|---|---|---|---|
| SYNTHETIC_1_200 | 89.33 ± 0.13 | 95.00 ± 0.19 | 98.39 ± 0.24 |
| SYNTHETIC_1_1000 | 87.82 ± 0.66 | 89.82 ± 0.42 | 95.01 ± 0.24 |
| SYNTHETIC_1_2000 | 81.80 ± 0.40 | 92.88 ± 0.36 | 96.05 ± 0.31 |
| SYNTHETIC_2_200 | 82.98 ± 0.10 | 92.48 ± 0.89 | 98.92 ± 0.39 |
| SYNTHETIC_2_1000 | 83.92 ± 0.55 | 87.75 ± 0.26 | 92.10 ± 0.79 |
| SYNTHETIC_2_2000 | 83.31± 0.19 | 86.75 ± 0.27 | 88.54 ± 0.67 |
| MUTAG | 85.79 ± 0.04 | 89.85 ± 0.22 | 92.37 ± 0.26 |
| PTC | 61.10 ± 0.19 | 63.6 ± 0.23 | 67.6 ± 0.25 |
| PROTEINS | 78.09 ± 0.04 | 80.7 ± 0.54 | 83.3 ± 0.06 |
| NCI1 | 75.6 ± 0.05 | 77.7 ± 0.03 | 80.3 ± 0.10 |
| IMDB-B | 69.24 ± 0.03 | 72.41 ± 0.12 | 72.5 ± 0.16 |
| COLLAB | 65.6 ± 0.02 | 69.4 ± 0.02 | 71.1 ± 0.16 |

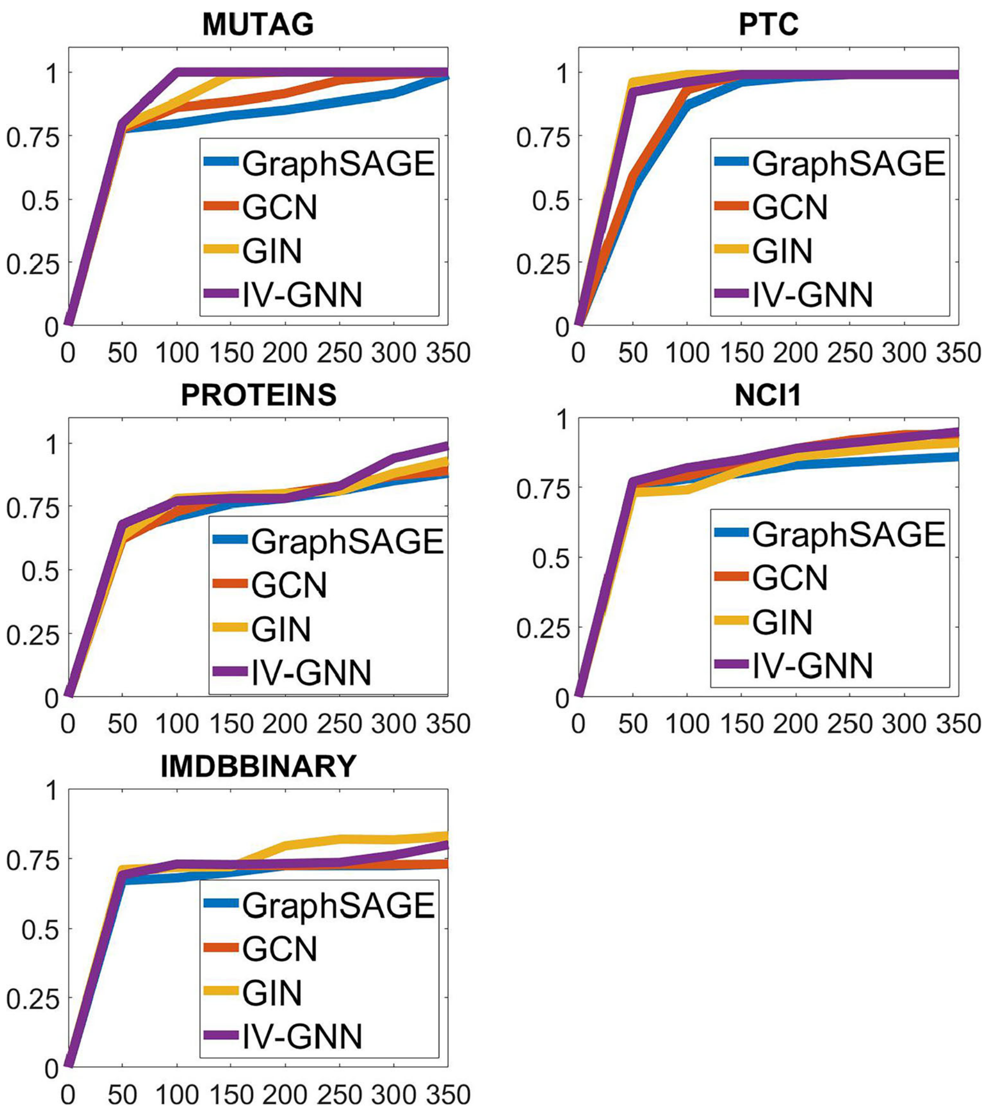

**Fig. 7** Training set performance of IV-GNN accepting degenerate interval value and other countable value feature accepting GNN. X-axis and Y-axis show the epoch number and the accuracy of training respectively

**Table 6** Test set classification accuracies in percentage

| Dataset | GraphSAGE | GCN | GIN | IV-GNN |
|---|---|---|---|---|
| MUTAG | $85.4 \pm 0.77$ | $82.9 \pm 0.66$ | $89.4 \pm 0.84$ | $94.7 \pm 0.24$ |
| PTC | $63.3 \pm 0.94$ | $66.9 \pm 0.19$ | $64.6 \pm 0.74$ | $68.5 \pm 0.34$ |
| PROTEINS | $75.8 \pm 0.34$ | $76.23 \pm 0.14$ | $76.75 \pm 0.98$ | $88.1 \pm 0.63$ |
| NCI1 | $78.1 \pm 0.34$ | $79.2 \pm 0.75$ | $82.5 \pm 0.11$ | $83.92 \pm 0.96$ |
| IMDB-B | $71.38 \pm 0.97$ | $72.41 \pm 0.9$ | $74.1 \pm 0.2$ | $73.35 \pm 0.68$ |
| COLLAB | - | - | $80.2 \pm 0.53$ | $79.85 \pm 0.3$ |

As recommended for GIN, we perform 10-fold cross-validation with LIB-SVM [7, 37] and perform the experiment for 350 epochs.

## 6.5 Results

### 6.5.1 Training set performance

We have already theoretically analyzed the representational power of our proposed IV-GNN. Now, we validate our theoretical findings by comparing training accuracies on various datasets. Ideally, the architecture, which has stronger representational power, should fit the data more accurately, resulting in better training performance. Training set performance gives an idea about how well the model learned from training data.

Figure 6 shows training curves of IV-GNN and interval-valued GNN with alternative aggregation functions with the same hyperparameter settings. In comparison, the GNN variants, which use less powerful interval aggregator as *AGGREGATE* function, cannot learn from many datasets. The reason behind this observation is, $agr_{new}$ has more distinguishing power than $agr_0$ and $agr_e$. Between $agr_0$ and $agr_e$, $agr_e$ performs better because as we have seen theoretically that, with non repetitive feature value, $agr_e$ is equally powerful as $agr_{new}$.

Figure 7 shows training curves of IV-GNN accepting degenerate interval valued feature and other state-of-the-art GNN models, who accept countable valued features. As we can see, curve of IV-GNN outgrows that of others, which proves that, IV-GNN learns from the data much better than other models in most of the cases. In cases of dataset IMDB-BINARY, GIN captures the dataset slightly better than IV-GNN. However, our proposed model IV-GNN is able to beat the other two models quite efficiently.

### 6.5.2 Test set performance

Now, we compare test set accuracies with respect to different interval aggregation-based GNN model and state-of-the-art GNN model. We have performed this experiment for the synthetic as well as real life datasets for 350 epochs and report the best cross-validation accuracy mean and standard deviation averaged over the 10 folds after performing each experiment *ten* times . In Table 5, we have seen that IV-GNN can capture graph structures and generalize well in most of the cases among other variants showing 4% better accuracy on an average.

In the special case for degenerate intervals, we perform experiments *ten* times on real-life datasets only and report the mean and standard deviation of the accuracies. We could not perform experiments of GraphSAGE and GCN on the dataset COLLAB due to memory bound. As we can see in the Table 6, IV-GNN performs much better than the other state-of-the-art approaches on four out of six datasets and achieves a performance gain of 7% on an average. On the other hand, for the datasets like IMDB-BINARY and COLLAB, IV-GNN's accuracy is not strictly the highest among GNN variants. However, IV-GNN is still comparable to the best performing GNN because a paired t-test at significance level 10% does not distinguish IV-GNN from the best.

## 6.6 Runtime comparison

We have experimented IV-GNN together with other three state-of-the-art approaches on five datasets (we could not include the dataset COLLAB for the experiment as GraphSAGE and GCN attain memory bound). As seen in the Fig. 8, IV-GNN and GIN take comparable amount of time. As expected from the theoretical findings, IV-GNN and GIN beat the other two models by utilizing the previously stored nodes' embeddings. However, GraphSAGE could not outperform GCN in these datasets, because the mini-batch based algorithm works better when the graph is very large.

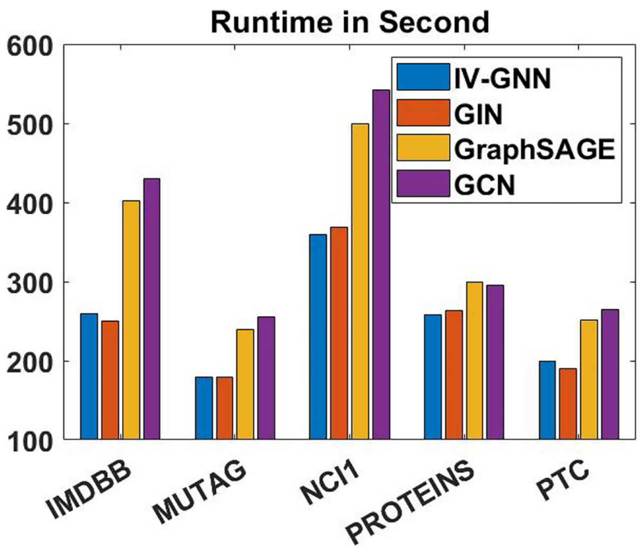

**Fig. 8** Comparison in runtime of different GNN models

## 6.7 Empirical study on hyperparameter setting

We have performed empirical study on various hyperparameters involved in the model and our experimental finding can be depicted in the Fig. 9.

### 6.7.1 Effect of different batch size

Batch size means the number of training examples to work through before updating the internal model parameters. We have experimented with the batch size {16, 32, 64, 128} and as we can see that for the relatively smaller dataset as MUTAG, PTC and PROTEINS, smaller batch size works better and to learn from the more enormous datasets such as COLLAB, NCI1 and IMDB-BINARY, larger batch size performs better. So we take batch size as {16, 64} depending on the size of the datasets.

### 6.7.2 Effect of different learning rate

Learning rate is a measure of the step size towards moving to the minimum of the loss function [27]. A lower learning rate can slow down the convergence process, while a too high learning rate may jump over the minima and oscillates. Based on the experimental result, we find that between {0.1, 0.01, 0.001}, 0.01 gives maximum accuracy for IV-GNN.

### 6.7.3 Effect of different hidden dimension

We have experimented with the different number of units in the hidden layer of the MLP, such as {16, 32, 64, 128}. Here also, smaller hidden dimension works well for smaller datasets as larger hidden dimension may cause underfitting. So, we take 32 as a hidden dimension for the bioinformatics

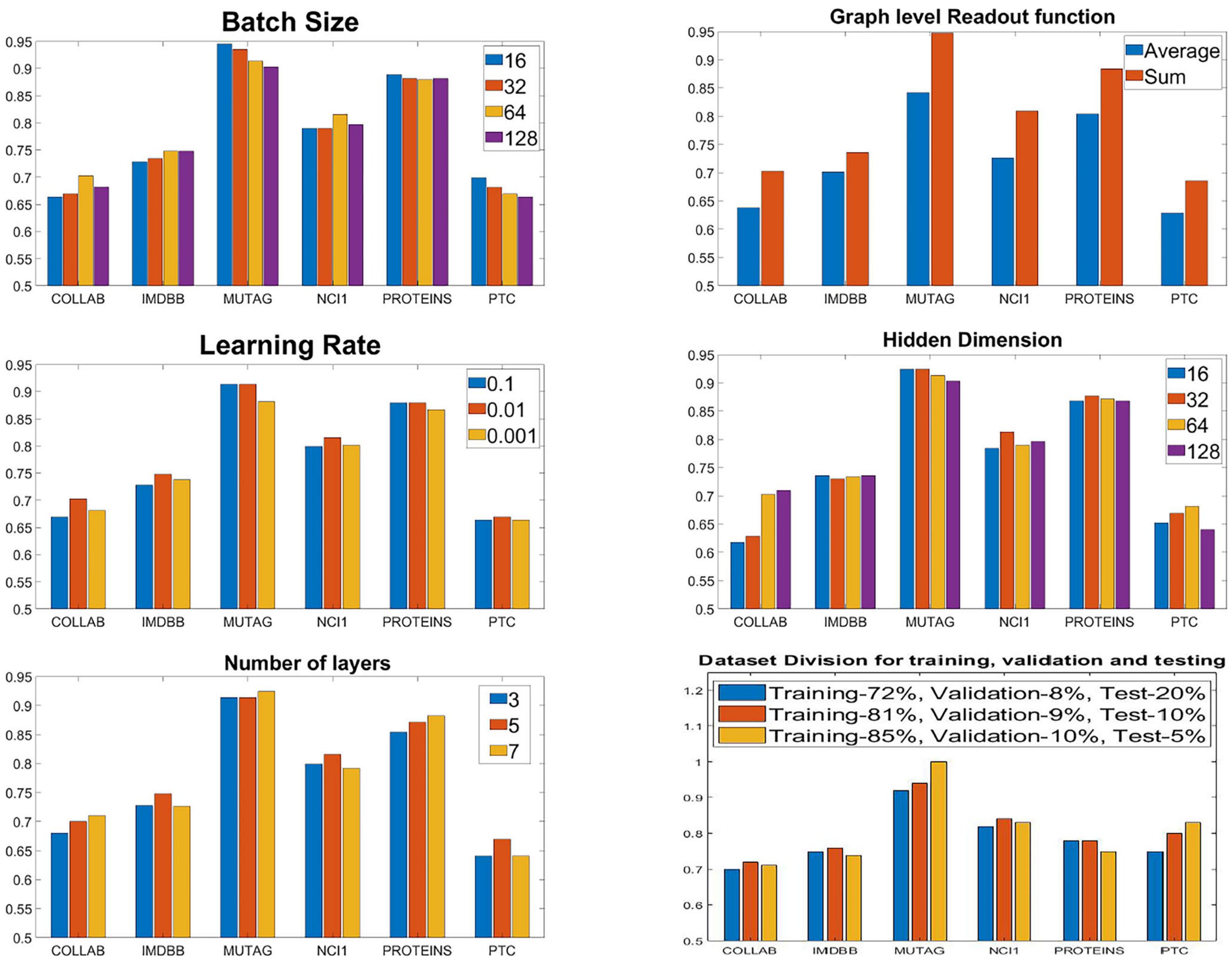

**Fig. 9** Empirical study on hyper-parameter setting. X-axis and Y-axis show the name of the dataset and the accuracy of training respectively

dataset. On the other hand, for the social network datasets, hidden dimension 128 captures the structures efficiently.

#### 6.7.4 Effect of different graph level readout function

We have experimented two different Graph level Readout functions *Sum* and *Average*. As shown in the figure, *Sum* performs much better than the function *Average*, which establishes the fact that *Sum* have better representational capacity than *Average*

#### 6.7.5 Effect of different number of GNN layers

We have tried our model with three different number of GNN layers (including the input layer), such as $\{3, 5, 7\}$. As we can see from the output, using 5 GNN layers in our model gives the best performance than others.

#### 6.7.6 Effect of different division of datasets

We have divided the datasets in three way as shown in the figure and found out that using $81\%, 9\%$ and $10\%$ of the datasets fortraining, validation and testing respectively fits best for most of the datasets.

## 7 Conclusion

This paper developed a new interval aggregation scheme, having much better discriminative power than existing aggregation function. Using this newly developed aggregation operator as AGGREGATE function, we develop a Graph Neural Network based architecture IV-GNN, which relaxes the condition on the feature space of being countable. Despite of being much more general in nature, the proposed method far outperforms the state-of-the-art approaches on several synthetic and real-life datasets(comparable results for IMDB-BINARY and COLLAB). Incidentally, the aggregation function, proposed in this work, is not continuous. Therefore, there may arise some situations where a slight change in aggregating intervals may bring significant change in the resultant interval, which is not expected. In future, we would like to apply this architecture to perform different graph-related tasks such as node classification, link prediction etc. Also, there are architectures having summarizing ability of a continuous data such as LSTM. We would like to include the interval-valued feature as a summarized embedding and investigate quality of the performance on graph classification task. Another exciting direction for future work is exploring different interval aggregation according to the demand of the situation. As application of our proposed model IV-GNN, an interesting area of research will be to perform prediction task on network time series data, where the node features are present in terms of time series data.

**Acknowledgements** We want to acknowledge support from J.C.Bose Fellowship[SB/S1/JCB-033/2016 to S.B.] by the DST, Govt. of India. Also, we would like to thank Dr. Monidipa Das and Dr. Monalisa Pal for their valuable comments.

## References

1. Ahn J, Peng M, Park C, Jeon Y (2012) A resampling approach for interval-valued data regression. Statistical Analysis and Data Mining: The ASA Data Science Journal 5(4):336–348
2. Billard L, Diday E (2000) Regression analysis for interval-valued data. In: Data analysis, classification, and related methods, pp 369–374. Springer
3. Billard L, Diday E (2006) Symbolic data analysis: Conceptual statistics and data mining John Wiley
4. Borgwardt KM, Ong CS, Schönauer S, Vishwanathan S, Smola AJ, Kriegel HP (2005) Protein function prediction via graph kernels. Bioinformatics 21(suppl_1):i47–i56
5. Bui KHN, Cho J, Yi H (2021) Spatial-temporal graph neural network for traffic forecasting: an overview and open research issues. Appl Intell, pp 1–12
6. Calvo T, Kolesárová A, Komorníková M, Mesiar R (2002) Aggregation operators: properties, classes and construction methods. In: Aggregation operators, pp 3–104. Springer
7. Chang CC, Lin CJ (2011) Libsvm: a library for support vector machines. ACM Trans Intell Syst Technol (TIST) 2(3):1–27
8. Chen L, Cui J, Tang X, Qian Y, Li Y, Zhang Y (2021) Rlpath: a knowledge graph link prediction method using reinforcement learning based attentive relation path searching and representation learning. Appl Intell, pp 1–12
9. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv:1406.1078
10. Dai H, Kozareva Z, Dai B, Smola A, Song L (2018) Learning steady-states of iterative algorithms over graphs. In: International conference on machine learning, pp 1106–1114
11. Debnath AK, Lopez De Compadre RL, Debnath G, Shusterman AJ, Hansch C (1991) Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. J Med Chem 34(2):786–797
12. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems, pp 3844–3852
13. Dutta S, Bedregal BR, Chakraborty MK (2015) Some instances of graded consequence in the context of interval-valued semantics. In: Indian conference on logic and its applications, pp 74–87. Springer
14. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp 1024–1034
15. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257
16. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
17. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907

18. Lacasa L, Luque B, Ballesteros F, Luque J, Nuno JC (2008) From time series to complex networks: the visibility graph. Proc Natl Acad Sci 105(13):4972–4975
19. Lawler EL (2001) Combinatorial optimization: networks and matroids courier corporation
20. Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pp 177–187
21. Li G, Semerci M, Yener B, Zaki MJ (2011) Graph classification via topological and label attributes. In: Proceedings of the 9th international workshop on mining and learning with graphs (MLG), San Diego, USA, vol 2
22. Li K, Feng Y, Gao Y, Qiu J (2020) Hierarchical graph attention networks for semi-supervised node classification. Appl Intell 50(10):3441–3451
23. Liu Q, Wang J, Network FTCRN (2011) With a hard-limiting activation function for constrained optimization with piecewise-linear objective functions. IEEE Trans Neural Netw 22(4):601–613
24. Liu Z, Zhou J (2020) Introduction to graph neural networks. Synth Lect Artif Intell Mach Learn 14(2):1–127
25. Llanas B, Lantarón S, Sáinz F. J (2008) Constructive approximation of discontinuous functions by neural networks. Neural Process Lett 27(3):209–226
26. Morris C, Ritzert M, Fey M, Hamilton WL, Lenssen JE, Rattan G, Grohe M (2019) Weisfeiler and leman go neural : higher-order graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 33, pp 4602–4609
27. Murphy KP (2012) Machine learning: a probabilistic perspective. MIT press
28. Neto EDAL, De Carvalho FA, Tenorio CP (2004) Univariate and multivariate linear regression methods to predict interval-valued features. In: Australasian joint conference on artificial intelligence, pp 526–537. Springer
29. Roque AMS, Maté C, Arroyo J, Sarabia Á (2007) Imlp : applying multi-layer perceptrons to interval-valued data. Neural Process Lett 25(2):157–169
30. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80
31. Shrivastava A, Li P (2014) A new space for comparing graphs. In: IEEE/ACM International conference on advances in social networks analysis and mining (ASONAM 2014), pp 62–71. IEEE
32. Toivonen H, Srinivasan A, King RD, Kramer S, Helma C (2003) Statistical evaluation of the predictive toxicology challenge 2000–2001. Bioinformatics 19(10):1183–1193
33. Wale N, Watson IA, Karypis G (2008) Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl Inf Syst 14(3):347–375
34. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems
35. Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks? arXiv:1810.00826
36. Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi KI, Jegelka S (2018) Representation learning on graphs with jumping knowledge networks. arXiv:1806.03536
37. Yanardag P, Vishwanathan S (2015) Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1365–1374
38. Yang F, Zhang H, Tao S (2021) Simplified multilayer graph convolutional networks with dropout. Appl Intell, pp 1–16
39. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 974–983

**Sucheta Dawn** is a Senior Research Fellow, currently working towards her Ph.D. degree from the Machine Intelligence Unit in Indian Statistical Institute, Kolkata, India. After completing B.Sc and M.Sc degrees in Mathematics, she received her M. Tech degree in Computer Science and Data Processing from Indian Institute of Technology, Kharagpur, India. Her research interests includeGraph Neural Network, Recommender System, Machine Learning, and Deep Learning.

**Sanghamitra Bandyopadhyay** received the Ph.D. degree in computer science from the Indian Statistical Institute (ISI), Kolkata, India, where she has been a Professor since 2007. She is currently the Director of ISI. She has authored or co-authored over 250 technical articles and has published five authored and edited books. Her current research interests include computational biology and bioinformatics, soft and evolutionary computation, pattern recognition, and data mining. Dr. Banyopadhyay is a fellow of the Indian National Science Academy, the National Academy of Sciences, India, and the Indian National Academy of Engineering, India. She is also fellow of IEEE, TWAS and IAPR. She is a recipient of several prestigious awards, including the Humboldt Fellowship from Germany, ICTP Senior Associate, Trieste, Italy, and the Shanti Swarup Bhatnagar Prize in engineering science.