Tackling Oversmoothing in GNN via Graph Sparsification: A Truss-based Approach

Tanvir Hossain*, Khaled Mohammed Saifuddin*, Muhammad Ifte Khairul Islam*, Farhan Tanvir*, Esra Akbas*
*Department of Computer Science, Georgia State University, Atlanta, GA 30302, USA
{thossain5, ksaifuddin1, mislam29}@student.gsu.edu, {ftanvir, eakbas1}@gsu.edu

***Abstract*—Graph Neural Network (GNN) achieves great success for node-level and graph-level tasks via encoding meaningful topological structures of networks in various domains, ranging from social to biological networks. However, repeated aggregation operations lead to excessive mixing of node representations, particularly in dense regions with multiple GNN layers, resulting in nearly indistinguishable embeddings. This phenomenon leads to the oversmoothing problem that hampers downstream graph analytics tasks. To overcome this issue, we propose a novel and flexible truss-based graph sparsification model that prunes edges from dense regions of the graph. Pruning redundant edges in dense regions helps to prevent the aggregation of excessive neighborhood information during hierarchical message passing and pooling in GNN models. We then utilize our sparsification model in the state-of-the-art baseline GNNs and pooling models, such as GIN, SAGPool, GMT, DiffPool, MinCutPool, HGP-SL, DMonPool, and AdamGNN. Extensive experiments on different real-world datasets show that our model significantly improves the performance of the baseline GNN models in the graph classification task.**

***Index Terms*—GNN, Oversmoothing, Graph Sparsification, k-truss Subgraphs, Graph Classification.**

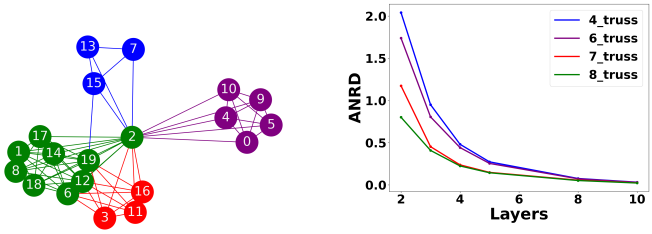
I. Introduction

In recent years, graph neural networks (GNN) have given promising performance in numerous applications over different domains, such as gene expression analysis [19], traffic flow forecasting [28], fraud detection [22], and recommendation system [7]. GNN effectively learns the representation of nodes and graphs via encoding topological graph structures into low-dimensional space through message passing and aggregation mechanisms. To learn the higher-order relations between nodes, especially for large graphs, we need to increase the number of layers. However, creating an expressive GNN model by adding more convolution layers increases redundant receptive fields for computational nodes and results in oversmoothing as node representations become nearly indistinguishable.

Several research works illustrate that due to oversmoothing, nodes lose their unique characteristics [5], [9], adversely affecting GNNs' performance on downstream tasks, including node and graph classification. Different models have been proposed to overcome the problem, such as skip connection [6], drop edge [17], GraphCON [18]. While many of these methods focus on node classification, they often overlook the impact of oversmoothing on the entire network's representation. Additionally, only a limited number of studies have investigated the influence of specific regions causing oversmoothing [6], [9] in GNNs. These studies show that the smoothness in GNN varies for complex connections in different graph areas, and an individual node with high degrees converges to stationary states earlier than lower-degree nodes. Hence, the networks' regional structures affect the phenomenon because repeated message passing occurs within the dense neighborhood regions of the nodes. Therefore, we observe the impact of congested graph regions on oversmoothing.

We conduct a small experiment to demonstrate the early oversmoothing at highly connected regions on a toy graph (Figure 1(a)). To calculate the density on the graph, we utilize the k-truss [1], one of the widely used cohesive subgraph extraction models based on the number of triangles each edge contains. To show the smoothness of the node features, we utilize the average node representation distance (ANRD) [11]. We measure the ANRD of different k-truss regions and present how it changes through the increasing number of layers in GNN. We present the toy graph and ANRD values with respect to the number of layers in Figure 1(b). While the toy graph in the figure is a 4-truss graph, it has 6, 7, and 8-truss subgraphs. Nodes and edges are colored based on their trussness. As known, k-truss subgraphs have hierarchical relations, e.g., 7 and 8-truss subgraphs are included in the 6-truss subgraph. Even at layer 2, we observe the ANRD of 7 and 8-truss subgraphs substantially degrades compared to the lower truss $(k = 4, 6)$ subgraphs.

While oversmoothing is observed at the node level, it may also result in losing crucial information for the graphs' representation to distinguish them. Furthermore, to learn the graph representation, GNNs employ various hierarchical pooling approaches, including hierarchical coarsening and message-



(a) Experiment Graph (b) ANRD vs Layers

Fig. 1: Early Oversmoothing: Average node representation distance(ANRD) of different k-truss subgraphs concerning GNN layers (2-10)

passing, resulting in oversmoothing via losing unique node features [24], [33]. Consequently, dense regions' identical node information affects the graph's representation learning.

We extend the preliminary investigation on the toy graph given in Figure 1(a). We first apply the SAGPool model. After each pooling layer's operation, we measure the coarsened graph's nodes' embedding space matrix (ESM) with l_2 norm, then present the results for the first 2 pooling layers in Figure 2(a) and 2(b). We observe that embedding distances are getting smaller for nodes within the dense regions, significantly reducing the final graph's representation variability. These node and graph representation characteristics through GNN models inspire us to work at different levels of dense regions in the network to mitigate oversmoothing.

Our Work. To tackle the challenge, we develop a truss-based graph sparsification (`TGS`) model. Earlier sparsification models apply supervised techniques [31] and randomly drop edges [9], [17] which may result in losing meaningful connections. However, our model selects the initial extraneous information propagating edges by utilizing edge trussness. It operates on the candidate edge's nodes' neighborhood and measures the connectivity strength of nodes. This connectivity strength assists in understanding the edge's local and global impact on GNN's propagation steps. Based on their specific node strength limits, we decide which edges to prune and which to keep. Removing selected redundant edges from dense regions reduces noisy message passing. That decreases oversmoothing and facilitates the GNN's consistency in performance during training and inference time. As we see in Figure 2(c) and 2(d), the sparsified graph exhibits greater diversity in node distances than the original, enhancing better representation learning. In a nutshell, the contributions of our model are listed as follows.

- We observe the prior stationary representation learning of nodes emerging in the network's high-truss region, which denotes a new perspective on explaining oversmoothing. We develop a unique truss-based graph sparsification technique to resolve this issue.
- In the edge pruning step, we measure the two nodes' average neighborhood trussness to detect the regional interconnectedness strength of the nodes. During the message passing steps in GNN, as we trim down the dense connections within subgraphs, nodes in less dense areas at varying hop distances acquire diverse hierarchical neighbor information. Conversely, nodes in highly dense regions receive reduced redundant information. This provides smoothness to the node representation as well as to the graph representation.
- We provide a simple but effective model by pruning noisy edges from graphs based on their nodes' average neighborhood trussness. The effectiveness of our model has been evaluated in comparison with standard GNN and graph pooling models. Extensive experiments on different real-world graphs show that our approach outperforms most of those baselines in graph classification tasks.

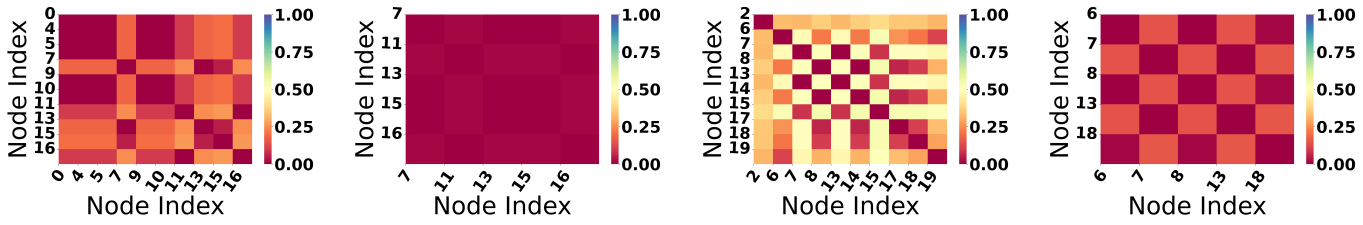
The rest of this paper is organized as follows. Section II discusses the related work that informs our research. Section III introduces the model's preliminaries, whereas Section IV describes the model itself. Next, Section V represents our model's experiment results and an analysis of its performance on different datasets. Finally, we conclude the paper with a discussion of future research directions.

II. Related Works

Graph Classification. Early GNN models leverage simple readout functions to embed the entire graph. GIN [26] introduces their lack of expressivity and employs deep multiset sums to represent graphs. In recent years, graph pooling methods have acquired excellent traction for graph representation. They consider essential nodes' features instead of all nodes. Flat pooling methods utilize the nodes' representation without considering their hierarchical structure. Among them, GMT [3] proposes a multiset transformer for capturing nodes' hierarchical interactions. Another approach, SOPool [25], capitalizes vertices second-order statistics for pooling graphs.

There are two main types of hierarchical pooling methods: clustering-based and selection-based. Clustering-based methods assign nodes to different clusters: computing a cluster assignment matrix [27], utilizing modularity [21] or spectral clustering [4] from the node's features and adjacency. On the other hand, selection-based models compute nodes' importance scores up to different hop neighbors and select essential nodes from them. Two notable methods are SAGPool [12], which employs a self-attention mechanism to compute the node importance, and HGP-SL [30], which uses a sparsemax function to pool graphs. KPLEXPOOL [2] hierarchically leverages k-plex and graph covers to capture essential graph structures and facilitates the diffusion between contexts of distance nodes. Some approaches combine both hierarchical pooling types to represent graphs. One model, ASAP [16], adapted a new self-attention mechanism for node sectioning, a convolution variant for cluster assignment. Another model, AdamGNN [32], employs multi-grained semantics for adapting selection and clustering for pooling graphs.

Oversmoothing: While increasing number of layers for a regular neural network may results better learning, it may cause an oversmoothing problem in which nodes get similar representations during graph learning because of the information propagation in GNN. To tackle this, researchers propose different approaches: DROPEDGE [17] randomly prunes edges like a data augmentor that reduces the message passing speed, DEGNN [14], [23] applies connectivity aware decompositions that balance information propagation flow and overfitting issue, MADGap [5] measures the average distance ratio between intra-class and inter-class nodes which lower value ensures over-smoothing. However, these methods overlook networks' regional impact on oversmoothing. The **k-truss** [10] algorithm primarily applies to community-based network operations to identify and extract various dense regions. It has been employed in different domains, such as high-performance computing [8] and graph compression [1].



(a) Org. pool 1 (b) Org. pool 2 (c) Spars. pool 1 (d) Spars. pool 2

Fig. 2: ESM of the toy graph and sparsified graph for $\delta = 7$

Our TGS model functions as a technique equipped with foundation graph pooling methods. It leverages the k-truss algorithm and edges' minimum node strength to provide networks' structural interconnectedness. Pruning highly dense connections helps to restrict excessive message passing paths to reduce oversmoothing in GNN models. We empirically justified it by experimenting with different graph topologies in section V.

III. Preliminaries

This section discusses the fundamental concepts for GNN and pooling, and also formulate the oversmoothing problem including the essential components for our solution to this problem. We begin with discussing graph neural networks and graph pooling techniques. Then, define the issue, including the task. Finally, we delve into the foundation concept of our model (k-truss), which plays a crucial role in solving the problem.

A. GNN and Graph Pooling

Graph Neural Network (GNN) [20] is an information processing framework that defines deep neural networks on graph data. Unlike traditional neural network architectures that excel in processing Euclidean data, GNNs are experts in handling non-Euclidean graph structure data. The principal purpose of GNN is to encode node, subgraph, and graph into low-dimension space that relies upon the graph's structure. In GNN, for each layer, K in the range $1, 2, ...k$, the computational node aggregates (1) messages $m^{(k)}_{N(v)}$ from its K-hop neighbors and updates (2) its representation $h_v^{(k+1)}$ with the help of the $AGGREGATE$ function.

$$m^{(k)}_{N(v)} = AGGREGATE^{(k)}(\{h_u^{(k)}, \forall u \in N(v)\}) \quad (1)$$

$$h_v^{(k+1)} = UPDATE^{(k)}(h_v^{(k)}, m^{(k)}_{N(v)}) \quad (2)$$

In the context of graph classification, GNNs must focus on aggregating and summarizing information across the entire graph. Hence, the pooling methods come into play.

Graph Pooling. [13] Graph pooling performs a crucial operation in encoding the entire graph into a compressed representation. This process is vital for graph classification tasks as it facilitates capturing the complex network structure into a meaningful form in low-dimensional vector space. During the nodes' representation learning process at different layers, one or more pooling function(s) operate on them. These pooling layers are pivotal in enhancing the network's ability to generalize from graph data through effective graph summarization. In general, pooling operations are categorized into two types: Flat pooling and Hierarchical pooling.

Flat pooling [13] is a straightforward graph readout operation. It simplifies the encoding by providing a uniform method to represent graphs of different sizes in a fixed size.

$$h_G = READOUT(\{h_v^{(k)} | v \in V\}) \quad (3)$$

Hierarchical pooling [13] iteratively coarsens the graph and encodes comprehensive information in each iteration, reducing the nodes and edges of the graph and preserving the encoding. It enables the graph's representations to achieve short and long-sighted structural details. In contrast to Flat Pooling, it gives deeper insights into inherent graph patterns and relationships.

Between the two types of hierarchical graph pooling methods, the *selection-based* methods emphasize prioritizing nodes by assigning them a score, aiming to retain the most significant nodes in the graph. They employ a particular $attention$ function for each node to compute the node importance. Based on the calculated scores, top k nodes are selected to construct a pooled graph. The following equation gives a general overview of the top k selection graph pooling method:

$$\begin{aligned} S = score(G, X);\ idx = topK(S, [\alpha \times N]) \\ A^{(l+1)} = A_{idx,idx} \end{aligned} \quad (4)$$

where $S \in \mathbb{R}^{N \times 1}$ is the scores of nodes, α is the pooling ratio, and N is the number of nodes. Conversely, *clustering-based* pooling methods form supernodes by grouping original graph nodes that summarize the original nodes' features. A cluster assignment matrix $S \in \mathbb{R}^{N \times K}$ using graph structure and/or node features are learned by the models. Then, nodes are merged into super nodes by $S \in \mathbb{R}^{N \times K}$ to construct the pooled graph at $(l+1)^{th}$ layer as follows

$$\begin{aligned} A^{(l+1)} = S^{(l)^T} A^{(l)} S^{(l)} \\ H^{(l+1)} = S^{(l)^T} H^{(l)} \end{aligned} \quad (5)$$

where $A \in \mathbb{R}^{\mathbb{N} \times \mathbb{N}}$ is the adjacency matrix and $H \in \mathbb{R}^{N \times d}$ is the feature matrix with d dimensional node feature and N is the number of nodes. Note that, the $AGGREGATE$,

$UPDATE$ and $READOUT$ operations are different operational functions, commonly including min, max, $average$, and $concat$.

B. Oversmoothing

According to [29], continual neighborhood aggregation of nodes' features gives an almost similar representation to nodes for an increasing number of layers K. simply, without considering the non-linear activation and transformation functions, the features converge as -

$$h^{\infty} = \hat{A}^{\infty} X, \quad \hat{A}_{i,j} = \frac{(d_i+1)^r (d_j+1)^{1-r}}{2m+n} \tag{6}$$

where, v_i and v_j are source and target nodes, d_i and d_j are their degrees respectively, $\hat{A}$ is the final smoothed adjacency matrix and $r \in [0,1]$ is the convolution coefficient. The equation (6) shows for an infinite number of propagations, the final features are blended and only rely upon the degrees of target and source nodes. Furthermore, through spectral and empirical analysis [6] shows: *nodes with higher-dree are more likely to suffer from oversmoothing*.

$$h^k(j) = \sqrt{d_j+1}\left(\sum_{i=1}^{n} \frac{\sqrt{d_j+1}}{2m+n} x_i \pm \frac{\sum_{i=1}^{n} x_i (1-\frac{\lambda_G^2}{2})^k}{\sqrt{d_j+1}}\right) \tag{7}$$

In the equation (7), λ_G is the spectral gap, m is the number of edges, and n is the number of nodes. It represents the features convergence relied upon the spectral gap λ_G and summation $\sum_{i=1}^{n}$ of feature entries. When the number of layers K goes to infinity, the second term disappears (after $\pm$). Hence, all vertices' features converge to steady-state for oversmoothing, which mainly depends on the nodes' degrees.

C. Problem Formulation

This research aims to alleviate oversmoothing by effectively simplifying graphs to balance global and local connections, resulting in better graph classification results. Formally, A Graph is denoted as $G = (V, E, X)$, where V is the set of nodes and E is the set of edges. Symbol $X \in \mathbb{R}^{N \times d}$ represents the graph's feature matrix of dimension d, where $N = |V|$ is the number of nodes in G and $x_v \in \mathbb{R}^d, x_v \in X \; and \; v \in V$ is a d dimensional feature of a particular node in the graph. The neighborhood of a node u is denoted as $N(u)$, and its degree is represented as $d(u) = |N(u)|$. For a dataset $D = (\mathbb{G}, Y)$ consisting of a set of graphs $\mathbb{G} = \{G_1, G_2 \cdots G_N\}$, label pair $Y = \{Y_1, Y_2, ...Y_N\}$, our truss-based sparsification algorithm introduces a set of sparsified graphs as $\mathbb{G}_S = \{G_{S1}, G_{S2}....G_{SN}\}$. The algorithm is designed to remove redundant graph connections and retain the graph's essential structural information. Subsequently, This sparsified graphs set is analyzed using GNN models to learn a function $f : \mathbb{G}_S \rightarrow Y$ leveraging the reduced complexity of the graphs. The principal objective is to enhance the accuracy (Acc) of GNN models in graph classification tasks.

D. K-truss

Identifying and extracting cohesive subgraphs is a pivotal task in the study of complex networks. The k-truss subgraph extraction algorithm is instrumental as it isolates subgraphs based on a specific connectivity criterion. The root of the criterion is the term $support$, which refers to the enumeration of triangles in which an edge participates. The $support$ serves as the cornerstone to measure the cohesiveness of a subgraph. The following two definitions explain the criterion for extracting specific tightly interconnected subgraphs from a complex network.

Definition 1: ***Support:*** *In graph $G = (V, E)$, the support of an edge $e = (u, v) \in E$ is denoted as $sup_G(e)$ the number of triangles where e involves, i.e $sup_G(e) = |\{ \Delta_{uvw} : w \in V \}|$.*

*Definition 2: k-**truss subgraph:** A subgraph $S = (V_S, E_s)$ where, $S \subseteq G$, $V_s \subseteq V$ and $E_S \subseteq E$ is a k-truss subgraph where every edge $e \in E_S$ has at least $k-2$ support, where $k \geq 2$.*

Notably, the concept of k-truss is inherently dependent on the count of triangles within the graph, establishing that any graph can be considered a 2-truss subgraph. The hierarchical structure of k-truss subgraphs implies that a 3-truss subgraph is a subset of the 2-truss subgraph (the original graph) denoted as $G_3 \subseteq G_2$. Similarly, $G_4 \subseteq G_3 \cdots G_k \subseteq G_{k-1}$.

Definition 3: ***Edge Trussness:*** For a given graph G, for $k > 2$, an edge, $E(u, v)$, can exist in multiple k-truss subgraphs. The trussness of the edge, denoted as $T_r(u, v)$, is quantified from the highest k value for which the edge is included in that subgraph. That is, $T_r(u, v) = k$ and $(u, v) \notin G_{k+1}$.

IV. Truss based Graph Sparsification

In this section, we explain the proposed truss-based graph sparsification model (TGS) to overcome the oversmoothing problem for graph classification. In graph analytics, classifying graphs is challenging due to their large size and complex structure. Graph sparsification– a technique that reduces the number of graph connections by preserving crucial graph structures, is an emerging technique to address these challenges. We aim to get an effective simplified graph that keeps essential short- and long-distance graph connections through graph sparsification, which produces the optimal graph classification result. The overall architecture of the proposed model is presented in Figure 3. The model consists of 2 parts: truss-based graph sparsification and graph learning on the sparsified graph. We observe four phases to develop the truss-based graph sparsification framework.

Phase 1: *Compute edge trussness:* At first, we apply the k-truss decomposition algorithm on an unweighted graph to compute its edges' trussness as weight. Next, we split all edges into groups based on their truss values: high-truss edges and low-truss values for a given threshold η.

Phase 2: *Measure node strenght:* TGS focuses on high-truss edges for sparsification. As high-truss edges have higher degrees, they massively contribute to the oversmoothing phenomenon (Section III-B). Thus, strategic pruning of those

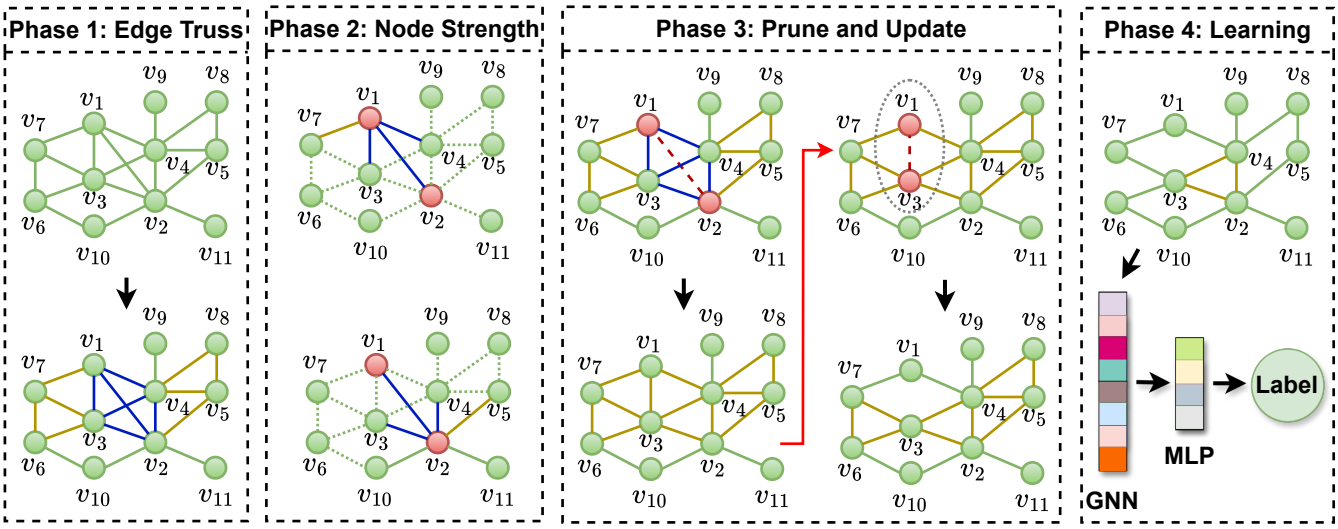


Fig. 3: Architecture of the TGS ($\eta = 3, \delta = 2.5$).

edges helps to reduce oversmoothing. However, at the same time, important structural connections need to be maintained. To do so, we measure the minimum node strength of its two end nodes for each candidate high-truss edge, indicating the edge's surroundings' density status.

Phase 3: *Prune and update:* When that minimum value exceeds the standard density assuring threshold, we prune the edge from the graph and update all the edge's trussness values. Due to the cascading effect of the network, pruning affects other edges' trussness. Therefore, edge trussness needs to be updated after each pruning operation. The process continues the pruning step until all high-truss edges are examined.

Phase 4: *Learning:* At the end of the sparsification, we first feed the processed graph to GNN models for graph learning. Finally, we experiment the entire graph's representation with a multi-layer perceptron (MLP) network.

Note that our model follows some strategies during the graph sparsification steps: (a) sorts the high-truss edges in descending order to prune more dense regions' edges earlier, and (b) examines each edge only once. Removing an edge from the graph might affect other edges; then, in further exploration, one edge might satisfy the pruning condition. The phenomenon negligibly happens as TGS starts to prune from more dense edges. Hence, the technique avoids recursion.

A. Dense Region Identification

To learn the structure of the graph, GNN applies message passing between nodes through edges. Through repeated message passing, nodes in the dense regions get similar neighbors' feature information, which causes oversmoothing. As a result, the features of those regions' nodes become indistinguishable. Many different density measures exist, including k-truss, k-core, and k-edge. This paper uses k-truss, defined based on triangle connectivity, to identify the dense regions.

Our approach employs a truss-decomposition [1] algorithm, as detailed in Algorithm 1, to compute edge trussness and

Algorithm 1: Computing Edge Trussness

Input: A Graph $G = (V, E)$
Output: A graph G_T, where edge trussness $T_r(e)$ as weight for each $e \in E$

1 Compute $sup_G(e)$, $\forall e \in E$
2 $Sort(e \in E,\ item = sup_G(e))$ // in non-decreasing order
3 $k \leftarrow 2$; $G_T \leftarrow G.copy()$
4 **while** $\exists e \in E, sup_G(e) \leq (k-2)$ **do**
5 $\quad e^*(u,v) \leftarrow argmin_{e \in E}\ sup_G(e)$ // assume w.o.l.g, $d(u) \leq d(v)$
6 $\quad$ **foreach** $w \in N(u) \cap N(v)$ ***and*** $e^* = (u,v) \in E$ **do**
7 $\quad\quad sup_G(u,w) \leftarrow sup_G(u,w) - 1$
8 $\quad\quad sup_G(v,w) \leftarrow sup_G(v,w) - 1$
9 $\quad\quad$ Reorder(u,w) and (v,w)
10 $\quad$ **end**
11 $\quad G_T[u][v][W] \leftarrow T_r(e^*) \leftarrow k$;
12 $\quad$ remove e^* from E
13 **end**
14 **if** $\exists e \in E$ **then**
15 $\quad k \leftarrow k + 1$
16 $\quad$ **goto** the while-loop (line 4)
17 **end**
18 **return** G_T

discover all k-trusses from G. At first, it takes an unweighted graph as input and computes the supports of all edges. Then initialize the value of k as 2 and select the edge e^* with the lowest support (line 5). Next, the value k is assigned as edge weight W, and the edge is removed (line 12). Removing an edge decreases other edges supports. Hence, we reorder edges according to their new support values (line 9). The process continues until the edges that have support no greater than

$(k-2)$ are removed from the graph. Next, the algorithm checks whether any edge exists to access or not. If one or more exist(s), it increments k by one and goes to the line 4 again to measure their trusseness (line 14-17). Edge trussness facilitates understanding the highest dense region within which the edge exists. After calculating the edge trussness, to identify highly dense areas, TGS separates the edges in G_T into two sets: High-Truss Edges and Low-Truss Edges. Following condition (1), it compares all edges' trussness with the given threshold value, η, and determines the High Truss Edges E_H. For example, in Figure 3, given $\eta = 3$, the blue ($T_r(E) = 4$) and golden ($T_r(E) = 3$) colored edges are high-truss edges. Pruning $\{E \in E_H\}$ reduces the load of high-degree nodes in dense regions, which assists in mitigating oversmoothing in GNN.

Condition 1: ***High Truss Edges*** E_H*: In any graph for a specific variable* η*, if an edge's trussness value is greater than or equal to* η *then the edge is considered as a high truss edge and their set is denoted as* E_H*.*

B. Pruning Redundant Edges

Ascertaining the high-truss edges is crucial for understanding the density level in different parts of the graph. However, directly pruning these edges may break up essential connectivity between nodes. For example, low-degree nodes could be connected with a dense region node, and pruning an incident high-truss edge may not provide adequate information to that low-degree node. To balance the connectivity between nodes, we determine the nodes' strength of edge high-truss edges and then proceed to the next step. To measure nodes' ($n \in E,\ E \in E_H$) strength, TGS calculates the average trussness $\bar{T}_{N(n)}, n \in V$. This score ensures the density depth of a node and implies its important connectivity.

Definition 4: The strength of a node is measured as the summation of all of its incident edge weights. However, In this research, **node strength** is applied as the average of nodes' incident edges' trussness.

$$\bar{T}_{N(n)} \leftarrow \frac{1}{|N(n)|} \sum_{u \in N(n)} T_r(n,u) \tag{8}$$

For a candidate edge $E = (u, v)$, after measuring the node strength of u and v (8) , their minimum value (9) has been taken. Notably, a node may be included in different k-truss subgraphs. Hence, its neighborhood's trussness provides more connectivity information. The minimum node strength of an edge's two endpoints signifies the least density of its surroundings. As we aim to reduce the density of highly connected regions to combat oversmoothing, TGS follows a technique to decide to prune edges. For this purpose, the minimum node strength of the edge E is compared to a threshold δ. In condition (2), This comparison ensures the edge's presence in a prunable dense region. The condition indicates that if any end of the candidate edge is sparse $\bar{T}_{N(E)} < \delta$, TGS avoids cutting it because that connection serves as an essential message-passing medium in the GNNs aggregation step. In contrast, when the minimum score equals or exceeds the value of δ, we assume the edge is part of a highly dense region, and there is a high chance of excessive messages passing between that region's nodes. That may cause them to blend their representations, leading to oversmoothing during graph learning through GNN (section III-B). From the condition, the model understands which edges contribute to undesirable density levels that foster oversmoothing in GNNs.

Algorithm 2: Truss-based Graph Sparsification (TGS)

Input: A Graph (G), threshold δ, cutoff η
Output: A Sparsified Graph $G_S \subset G$

```
19 G_T = Computing Edge Trussness(G)
20 E_H = {∀(u,v) ∈ E, T_r(u,v) ≥ η}
21 E_H ← Sort(E_H, reverse = True)
22 n ← length(E_H)
23 G_S ← G_T
24 for r ← 1 to n do
25     foreach E(u,v) ∈ E_H do
26         Compute T̄_{N(E)} from Equations (8) and (9)
27         if T̄_{N(E)} ≥ δ then
28             G_S ← G_T \ E(u,v) // Edge Cut
29             E_H ← E_H \ E(u,v) // Reduce High Truss
                Edge List
30             G_S ← UpdateTr(G_S)
31         end
32     end
33 end
34 return G_S
```

$$\bar{T}_{N(E)} \leftarrow minimum(\bar{T}_{N(u)}, \bar{T}_{N(v)}) \tag{9}$$

Condition 2: An Edge $e = (u, v)$ is eligible for pruning when the minimum average neighborhood edge weight between u and v equals or exceeds the threshold δ.

$$\bar{T}_{N(E)} \geq \delta \tag{10}$$

For example, at the lower-left in phase 3 (in Figure 3), the edge (v_2, v_{10}), where the degrees of v_2 and v_{10} are 5 and 2, respectively. The node strengths, $\bar{T}_{N(v_2)}$ is $\{(3 \times 3) + (2 \times 2)\}/5 = 2.6$, and $\bar{T}_{N(v_{10})}$ is $\{(2+2)\}/2 = 2$. Given $\delta = 2.5$, and the minimum node strength, $\bar{T}_{N(v_2,v_{10})} = \bar{T}_{N(E)} = min\ (2.6,\ 2) = 2$. Hence, the pruning condition is unsatisfactory, and the edge will stand in the graph. If TGS pruned the edge, the neighborhood of v_{10} would be sparser than before and miss its crucial global information. On the other hand, at the upper-right in phase 3, in context of $E = (v_1, v_3)$, $\bar{T}_{N(v_1)} = 3$ and $\bar{T}_{N(v_3)} = 3$. Hence, comparing to the value of δ they are already in the dense region and $\bar{T}_{N(v_1,v_3)} = 3 \geq 2.5$. In this case, the pruning will help to prevent blended node representation in GNN, especially between highly interconnected subgraphs. According to our model, it considers the nodes will still stay in enough dense regions to receive meaningful local and global neighborhood information after pruning.

TABLE I: Datasets' Statistics

Datasets	# Graphs	Avg # $\|V\|$	Avg # $\|E\|$	# Classes
PROTEINS	1113	39.06	72.82	2
NCI1	4110	29.87	32.30	2
NCI109	4127	29.68	32.13	2
PTC	344	25.56	25.96	2
DD	1178	284.32	715.66	2
IMDB-B	1000	19.77	96.53	2
IMDB-M	1500	13.00	65.94	3
REDDIT-B	2000	429.63	497.75	2

The Algorithm 2 represents the TGS model. Lines (19-21) identify the dense regions and ensure high-truss edges of the network while lines (24-33) demonstrate the pruning of noisy high-truss edges in the network. Algorithm $UpdateTr$ in line 30 (similar to section 4.2, [10]), updates all edges' trussness after each pruning step.

C. Algorithm Complexity

The complexity of measuring edge trussness is $O(E\sqrt(E))$ and the updateTr algorithms complexity is O(E). As we explore all high truss edges in the algorithm, the exploration complexity is O(E) in the worst case. Hence, the Algorithms complexity is $O(E(O(E)) + O(E\sqrt(E)) = O(E^2) + O(E\sqrt(E))$ = $O(E^2)$ in the worst case. Although it seems highly complex the real-world datasets are not hardly dense. In addition, due to updating the edges trussness score many high-truss edges are removed before examination.

V. Experiment Design and Analysis

This section validates our technique on different real-world datasets by applying standard graph pooling models. First, we provide an overview of the datasets. Then, we briefly describe the parameters of various methods. Finally, we compare the performance of our TGS algorithm's enhancement with the original baselines in the graph classification tasks including analysis of parameters, deeper networks, and ablation study.

A. Datasets and Baselines

We experiment with our model on eight different TU Dortmund [15] datasets: Five of them are biomedical domain: **PROT**EINS, **NCI1, NCI109, PTC, and DD** and three of them from social network domain: **IMDB-B**INARY, **IMDB-M**ULTI, and **REDDIT-B**INARY. We extend the TGS algorithm with seven state-of-the-art backbone graph pooling models. Among them, three are node clustering-based pooling methods: **Diff**Pool [27], **DMon**Pool [21] and **MinCut**Pool [4]. Two models, **SAG**Pool [12] and **HGP-SL [30]**, utilize a node selection approach for pooling the graphs. Of the remaining two, one learns graph representation through flat-pooling (**GMT [3]**), and another one utilizes an adaptive pooling approach by applying both node selection and clustering for the pooling procedure: **Ad**am**G**NN [32]. We report the statistics of the datasets in the Table I.

B. Experimental Settings

To compare fairly, we executed the existing standard implementations of the baselines and incorporated them with our model. For evaluation, we split the datasets into 80% for training, 10% for validation, and 10% testing. Mostly, we stopped learning early for 50 consecutive same validation results in training. We measured the performance using the accuracy metric by ruining each model 10 times for 10 random seeds and reported their mean. The batch size was 128 for most of the models. The effectiveness of our pruning method mostly depends on two crucial parameters: the cutoff parameter η and the edge pruning threshold δ. For all experiments, we set $\eta = 3$, which means any edge with a trussness score below 3 cannot be pruned from the graph. On the other hand, we experimented with various δ values across the datasets. Specifically, we used δ values of $\{3, 4, 5, 6, 7\}$ for IMDB-BINARY, IMDB-MULTI, and $\{3, 3.5, 4\}$ for REDDIT-BINARY datasets. For PROTEINS and DD datasets δ was set as $\{3, 3.25, 3.5, 3.75, \ and \ 4\}$ and for NCI1, NCI109, we used δ values of $\{$ 2.5, and $3\}$ while for PTC only 2.5.

C. Result Analysis

Table II reports the experiment results, providing a comparative analysis between our established model and original baselines across various datasets. TGS integrated with backbone graph pooling models consistently outperforms the baselines, and demonstrates its robustness in graph classification tasks. On selection-based models, with the incorporation of the SAGPool model, TGS achieves a 1.5-5.5% gain($\mathbb{G}$) (11) over the original models. Notably, on DD and IMDB-BINARY datasets, the gains are 3.34% and 5.17%, respectively. In the experiment with the HGP-SL model, TGS attains a sustainable improvement of nearly 4.5% on the IMDB-BINARY dataset and on the PTC dataset, which is over 2.5%. Adapting TGS along with the flat pooling model GMT acquires a significant gain (nearly 7%) over the NCI109 dataset and maintains consistent performance on other datasets.

In experiments with cluster-based modes, TGS equipped to Diffpool model achieves a magnificent accuracy gain on the PTC dataset, which is nearly 19%. It also demonstrates strong performance with the DMonPool model over all datasets. Notably, it achieves the highest accuracy on the REDDIT-BINARY dataset, which is 85.75%

$$Gain(\mathbb{G}) = \frac{\texttt{TGS} - \text{Original}}{\text{Original}} \times 100\% \tag{11}$$

Extended Experiment: In addition to the pooling method, we incorporate the TGS model with the fundamental GNN models: two versions of graph isomorphic networks (GIN-0 and GIN-ϵ) and the simple graph convolution network for graph classification. During the experiment with GIN networks, we follow 10-fold cross-validation to evaluate the validity of our model. On the other hand, we assess the GCN as other pooling models (section V-B). In most contexts (in Table II), our technique outperforms these models on every dataset. Especially on the PTC and IMDB-BINARY

TABLE II: Result Table (in %). Results that achieve at least 0.5% gain over the counterpart model we mark in bold. We show a model-by-model comparison.

Backbones		Biomedical Dataset					Social Network Dataset		
		PROTEINS	**NCI1**	**NCI109**	**PTC**	**DD**	**IMDB-B**	**IMDB-M**	**REDDIT-B**
SAGPool	Original	72.68	70.10	67.37	57.14	77.31	71.60	44.87	77.55
	TGS	**73.84**	**70.72**	**67.78**	**58.00**	**80.67**	**75.30**	44.67	**79.05**
GMT	Original	70.63	60.29	48.86	50.86	65.88	71.10	47.40	70.95
	TGS	**71.25**	**61.56**	**52.32**	51.14	**69.50**	**74.50**	**47.93**	71.25
DiffPool	Original	**71.10**	66.96	67.60	46.29	75.31	63.90	**44.80**	80.92
	TGS	69.82	**68.30**	67.20	**55.14**	**78.75**	**64.80**	42.67	**82.46**
DMon	Original	75.45	74.20	72.61	53.71	79.32	74.00	49.53	84.95
	TGS	75.80	**74.74**	**73.77**	**56.00**	**80.42**	**74.90**	49.60	**85.85**
MinCut	Original	73.75	72.77	73.28	56.28	76.63	71.40	51.06	76.85
	TGS	73.84	**73.58**	73.04	**58.57**	**77.31**	**72.80**	51.20	77.05
AdamGNN	Original	78.12	47.36	65.16	60.00	70.63	72.48	49.53	OOM
	TGS	**81.77**	47.36	**67.81**	**62.86**	**74.25**	**77.60**	**50.57**	OOM
HGP-SL	Original	74.64	73.33	72.87	56.00	72.35	72.90	49.47	OOM
	TGS	74.28	73.38	**74.22**	**57.43**	**73.19**	**76.10**	**49.80**	OOM
GIN-0	Original	73.42	81.70	74.99	68.86	74.58	73.00	47.60	73.60
	TGS	73.84	**82.50**	75.08	**69.43**	**75.93**	**78.10**	**52.53**	**74.35**
GIN-e	Original	73.12	81.85	75.93	68.28	**76.44**	73.80	49.33	73.55
	TGS	73.39	**82.50**	**76.97**	68.29	74.75	73.90	**53.40**	**74.55**
GCN	Original	68.29	71.41	69.61	52.00	64.87	75.20	50.00	82.30
	TGS	**70.45**	**72.24**	69.69	**52.57**	**73.10**	75.50	**50.60**	**84.30**

datasets, TGS(GIN-0) achieves the highest accuracy scores of 69.43% and 78.10%, respectively. Additionally, TGS with the backbone GCN model, attains the overall second-highest accuracy on the REDDIT-BINARY data, with 84.30%.

D. Analysis in Deeper Network

We examine the impact of the TGS in deeper layers with the backbone models. In this analysis along with the SAGPool, we choose three GNN models: GCN, GIN-ϵ, and GIN-0. Besides, we select two datasets from the biomedical domain (DD and PROTEINS) and one from the social network domain (IMDB-BINARY). Figure 4 shows the best two ranked (Table IV and V in the section VII) TGS variants for the threshold δ compared to the original model performance. Columns 1 (TGS(GCN)) and 4 (TGS(SAGPool)) reveal that for increasing the number of layers, in most cases the TGS outperforms the original models on all three datasets in multiple layers. Figure 4(j) and 4(k) illustrate the similar trends in the context of TGS(GIN-ϵ) and TGS(GIN-0) on the IMDB-BINARY dataset. However, both of these models show fluctuations in accuracy on the DD and PROTEINS datasets. One interesting fact is that the accuracy trend TGS(GIN-0) has dis-proportionally increased in deeper networks on DD. A possible reason could be the dense nature of the networks in the dataset.

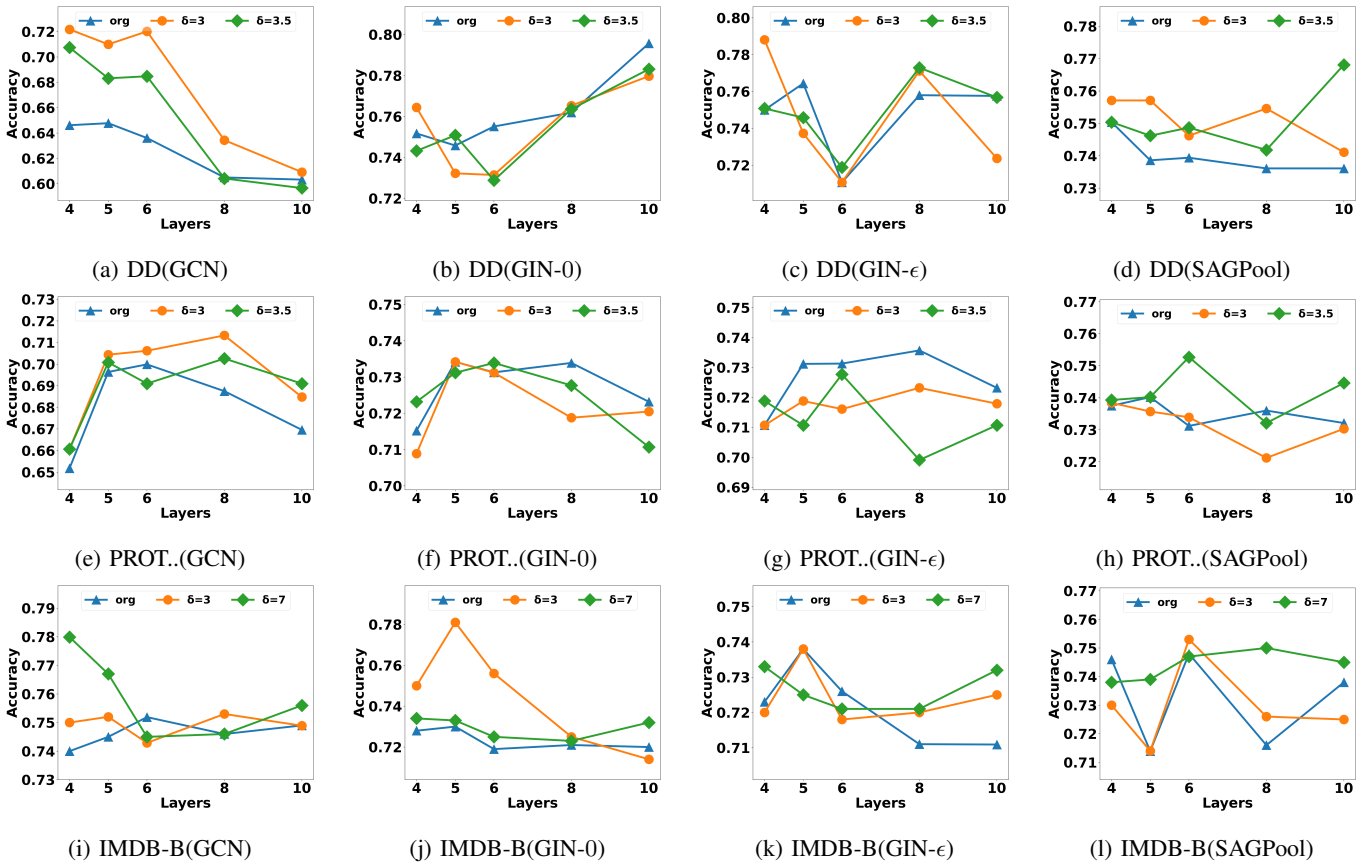
E. Sensitivity Analysis

In Figure 5, we demonstrate our model's performance for variations of hyperparameters' values on the IMDB-BINARY and PROTEINS datasets. Notably, in most cases, the pruning rate decreases as much as the delta value increases. Figure 5(a) shows at $\delta = 3$ value, our equipped TGS models perform well. We observe that for lower δ value, the accuracy of TGS with AdamGNN increases, whereas degrades for TGS(SAGPool). On the other hand, in the PROTEINS dataset (Figure 5(c)), for changing the δ value, TGS decorated with SAGPool, MinCutPool, DMonPool, and HGP-SL, showing near-consistent performance. However, when increasing the δ value from 3 to 3.25, TGS(AdamGNN)'s accuracy decreases and shows an almost stable performance. Assembled with GMT and Diffpool, the accuracy of TGS shows some variations from $\delta = (3 - 3.5)$ and then remains near the same score for other values.

Regarding the change of cutoff variable η, figures 5(b) and 5(d) show the performance changes on the same datasets. Similar to δ, the number of pruned edges in the graph decreases for increasing the value η, and TGS(SAGPool)'s accuracy increases. In contrast, for the same reason, the performance of our skill with backbone AdamGNN and HGP-SL degrades. Other models display minor fluctuations in accuracy with the change of η value.

F. Ablation Study

This section observes the strategical and functional significance of TGS with the backbone graph pooling models. We chose four datasets and six pooling methods. In Table III, for each dataset at the first two rows, we change the edge's connectivity strength measuring equations (8), and (9). In the first row, the equation (8) remains the same but at equation (9) instead of $minimum$ the $average$ of two end nodes' strength has been taken. On the other hand, in the second row, node strength measuring equation (8) is modified whereas the other equation remains unchanged. In the last two rows, we change the pruning procedure, examining to prune two (prune 2^*) and three (prune 3^*) edges without updating the edge trussness. Due to the change of equations, the model's performance on the PROTEINS dataset increases with its extension to MinCutPool and DiffPool methods. However, on the NCI1 dataset, the performance of AdamGNN dramatically decreases (47.36% and 61.22%). Regarding examining 2 and 3 edges for pruning, sometimes more than one edge is pruned without updating the

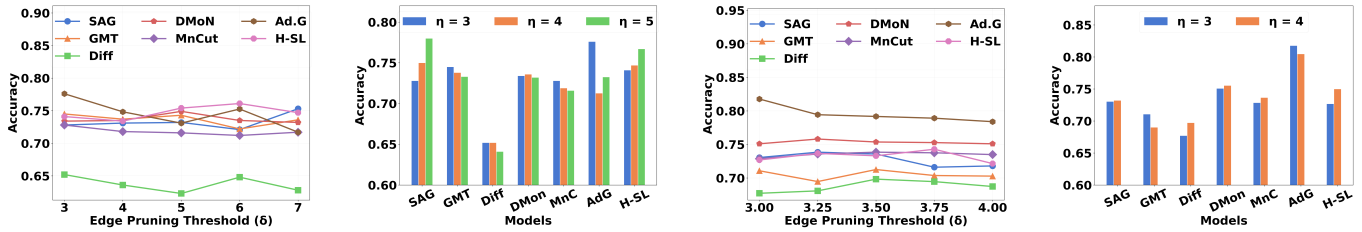


(a) DD(GCN) (b) DD(GIN-0) (c) DD(GIN-ϵ) (d) DD(SAGPool)

(e) PROT..(GCN) (f) PROT..(GIN-0) (g) PROT..(GIN-ϵ) (h) PROT..(SAGPool)

(i) IMDB-B(GCN) (j) IMDB-B(GIN-0) (k) IMDB-B(GIN-ϵ) (l) IMDB-B(SAGPool)

Fig. 4: Models' Performance in Deeper Networks



(a) IMDB-B δ (b) IMDB-B η (c) PROTEINS δ (d) PROTEINS η

Fig. 5: Change of parameters δ and η on IMDB-BINARY and PROTEINS.

other edges' trussness. Hence, significant information processing connections could prune in the system. Compared to the result in Table II, our model's performance severely degrades on some datasets with the components change. Nonetheless, the modified pruning strategy with MinCutPool and DiffPool achieve better results over TGS are 74.37% and 70.27% respectively on the PROTEINS dataset.

VI. Conclusion

In this paper, we have proposed an effective k-truss-based graph sparsification model to facilitate graph learning of the graph neural networks (GNN). Through the sparsification of dense graph regions' overflowed message passing edges, our model includes more variability to the input graph for alleviating oversmoothing. Comprehensive experiments on eight renowned datasets verify that TGS is consistent in performance over popular graph pooling and readout-based GNN models. We expect our research to show some interesting directions: Learning the edge pruning threshold during training, applying parallelization during pruning edges at different k-truss subgraphs, and joint learning to measure edge importance during graph sparsification.

References

[1] E. Akbas and P. Zhao. Truss-based community search: a truss-equivalence based indexing approach. *Proceedings of the VLDB Endowment*, 10(11):1298–1309, 2017.

TABLE III: Ablation Study Table. Boldly marked entries perform better than TGS.

Changes		SAGPool	GMT	DMonPool	MnCutPool	DiffPool	AdamGNN
IMDB-BINARY	avg (avg, avg)	75.00	**74.59**	74.70	**73.09**	63.00	75.43
	min (min, min)	74.00	48.06	72.70	**73.13**	**65.10**	**79.77**
	Prune 2*	70.99	48.69	51.80	48.50	61.10	53.73
	Prune 3*	68.70	47.90	54.89	49.59	62.79	59.80
IMDB-MULTI	avg (avg, avg)	44.67	47.40	48.86	50.59	41.46	50.04
	min (min, min)	46.67	**48.06**	49.13	50.73	38.86	46.92
	Prune 2*	43.99	47.93	49.73	49.09	39.47	50.04
	Prune 3*	**50.59**	47.46	49.40	50.99	41.06	47.96
PROTEINS	avg (avg, avg)	71.14	70.62	**75.53**	**74.01**	**69.99**	81.51
	min (min, min)	69.64	67.41	73.83	**74.10**	**71.01**	74.10
	Prune 2*	72.40	66.78	73.84	**74.37**	**70.27**	69.79
	Prune 3*	72.58	66.60	59.46	62.05	69.54	71.35
NCI1	avg (avg, avg)	70.07	61.55	68.8	69.97	68.10	47.36
	min (min, min)	**73.47**	61.19	69.05	73.52	68.17	61.22
	Prune 2*	67.73	60.97	69.63	70.34	68.17	58.16
	Prune 3*	68.71	59.17	68.83	70.07	67.15	60.02

[2] D. Bacciu, A. Conte, R. Grossi, F. Landolfi, and A. Marino. K-plex cover pooling for graph neural networks. *Data Mining and Knowledge Discovery*, 35(5):2200–2220, 2021.

[3] J. Baek, M. Kang, and S. J. Hwang. Accurate learning of graph representations with graph multiset pooling. *arXiv preprint arXiv:2102.11533*, 2021.

[4] F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*, pages 874–883. PMLR, 2020.

[5] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[6] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pages 1725–1735. PMLR, 2020.

[7] Y. Deng. Recommender systems based on graph embedding techniques: A review. *IEEE Access*, 10:51587–51633, 2022.

[8] S. Diab, M. G. Olabi, and I. El Hajj. Ktrussexplorer: Exploring the design space of k-truss decomposition optimizations on gpus. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–8. IEEE, 2020.

[9] R. Huang and P. Li. Hub-hub connections matter: Improving edge dropout to relieve over-smoothing in graph neural networks. *Knowledge-Based Systems*, 270:110556, 2023.

[10] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1311–1322, 2014.

[11] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[12] J. Lee, I. Lee, and J. Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR, 2019.

[13] C. Liu, Y. Zhan, J. Wu, C. Li, B. Du, W. Hu, T. Liu, and D. Tao. Graph pooling for graph neural networks: Progress, challenges, and opportunities. *arXiv preprint arXiv:2204.07321*, 2022.

[14] X. Miao, N. M. Gürel, W. Zhang, Z. Han, B. Li, W. Min, S. X. Rao, H. Ren, Y. Shan, Y. Shao, et al. Degnn: Improving graph neural networks with graph decomposition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1223–1233, 2021.

[15] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[16] E. Ranjan, S. Sanyal, and P. Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5470–5477, 2020.

[17] Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[18] T. K. Rusch, B. Chamberlain, J. Rowbottom, S. Mishra, and M. Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022.

[19] K. M. Saifuddin, B. Bumgardner, F. Tanvir, and E. Akbas. Hygnn: Drug-drug interaction prediction via hypergraph neural network. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1503–1516. IEEE, 2023.

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[21] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

[22] R. Van Belle, C. Van Damme, H. Tytgat, and J. De Weerdt. Inductive graph representation learning for fraud detection. *Expert Systems with Applications*, 193:116463, 2022.

[23] Y. Wang and T. Derr. Tree decomposed graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2040–2049, 2021.

[24] Y. Wang, H. Wang, H. Jin, X. Huang, and X. Wang. Exploring graph capsual network for graph classification. *Information Sciences*, 581:932–950, 2021.

[25] Z. Wang and S. Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[26] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[27] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[28] S. Zhang, Y. Guo, P. Zhao, C. Zheng, and X. Chen. A graph-based temporal attention framework for multi-sensor traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7743–7758, 2021.

[29] W. Zhang, M. Yang, Z. Sheng, Y. Li, W. Ouyang, Y. Tao, Z. Yang, and B. Cui. Node dependent local smoothing for scalable graph learning. *Advances in Neural Information Processing Systems*, 34:20321–20332, 2021.

[30] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, and C. Wang. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954*, 2019.

[31] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020.

[32] Z. Zhong, C.-T. Li, and J. Pang. Multi-grained semantics-aware graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[33] X. Zuo, H. Yuan, B. Yang, H. Wang, and Y. Wang. Exploring graph capsual network and graphormer for graph classification. *Information Sciences*, 640:119045, 2023.

VII. SUPPLEMENT

A. Result Details

This section reports the experiment details of the TGS model pipelined with the backbone graph pooling and GNN models. In all the tables, the social networks and biomedical domains' datasets' results are shown with the accuracy (%) metric. We measure the average accuracy for each dataset and rank them for different variations of the edge pruning threshold δ compared to the original backbone model's scores. Table IV and V report all the results of different TGS-variants for separate δ values. Due to limited space, we represent the (REDDIT-BINARY & PTC) and (NCI1 & NCI109) datasets' results together in sub-tables.

TABLE IV: IMDB-B, IMDB-M, REDDIT-B and PTC results

IMDB-BINARY						
		δ				
	Original	**3**	**4**	**5**	**6**	**7**
SAGPool	72.80	73.10	73.10	73.20	72.10	75.30
GMT	71.10	74.50	73.70	72.20	72.20	73.60
DiffPool	63.90	65.20	63.60	62.30	64.80	62.80
DMon	74.00	73.40	73.50	74.90	73.50	73.20
MinCut	71.40	72.80	71.80	71.60	71.20	71.70
AdamGNN	72.48	77.60	74.83	73.09	75.26	71.70
HGP-SL	72.90	74.10	73.40	75.40	76.10	74.70
GIN-ϵ	73.80	73.80	73.90	73.00	73.40	72.50
GIN-0	73.00	78.10	69.50	71.60	73.20	73.30
GCN	74.00	75.00	79.00	78.99	74.00	77.99
Mean	71.94	73.76	72.63	72.62	72.58	72.68
Rank	6	1	3	4	5	2

IMDB-MULTI						
		δ				
	Original	**3**	**4**	**5**	**6**	**7**
SAGPool	44.86	42.67	44.67	41.47	42.73	43.00
GMT	47.40	47.60	45.53	46.27	47.93	47.13
DiffPool	44.80	41.53	40.93	40.67	40.67	42.67
DMon	49.53	49.30	48.4	48.46	49.60	48.33
MinCut	51.06	50.06	51.20	50.73	50.20	50.93
AdamGNN	49.53	46.92	46.40	50.57	50.05	46.40
HGP-SL	49.46	49.80	48.80	48.87	47.33	49.00
GIN-ϵ	49.33	43.80	49.00	47.93	50.80	53.40
GIN-0	47.60	41.47	46.13	48.73	51.00	52.53
GCN	41.33	44.67	40.67	42.67	46.67	40.67
Mean	47.49	45.78	46.17	46.64	47.70	47.41
Rank	2	6	5	4	1	3

	REDDIT-BINARY				**PTC**	
		δ				δ
	Org.	**3**	**3.5**	**4**	**Org.**	**2.5**
SAGPool	77.55	76.15	79.05	77.80	57.14	59.14
GMT	70.95	70.95	71.05	71.25	50.86	51.14
DiffPool	80.72	81.45	82.32	82.03	46.29	55.14
DMon	84.95	85.75	84.70	85.85	53.71	56.00
MinCut	76.85	77.05	76.19	76.45	56.28	58.57
AdamGNN	OOM	OOM	OOM	OOM	60.00	62.86
HGP-SL	OOM	OOM	OOM	OOM	56.00	57.43
GIN-ϵ	73.55	73.75	74.00	74.55	68.28	66.57
GIN-0	73.60	73.65	74.10	73.30	68.86	65.43
GCN	88.99	89.50	88.50	73.50	45.71	45.71
Mean	78.40	78.53	78.74	76.84	56.31	57.80
Rank	3	2	1	4	2	1

TABLE V: DD, PROTEINS, NCI1 and NCI109 results

DD						
		δ				
	Original	**3**	**3.25**	**3.5**	**3.75**	**4**
SAGPool	77.31	79.83	78.15	80.67	80.67	79.00
GMT	65.88	69.5	67.39	68.49	64.62	63.78
DiffPool	75.31	78.13	78.75	78.13	77.81	73.75
DMon	79.32	80.42	79.49	79.57	79.66	80.08
MinCut	76.63	77.31	77.22	76.97	76.55	76.72
AdamGNN	64.63	71.65	74.25	71.12	67.87	71.00
HGP-SL	72.35	71.68	72.44	71.43	72.86	73.19
GIN-ϵ	76.44	73.73	71.36	74.58	73.9	74.75
GIN-0	74.58	73.22	74.75	75.08	74.49	75.93
GCN	58.82	73.95	66.38	67.23	70.58	69.75
Mean	72.13	74.94	74.02	74.33	73.90	73.80
Rank	6	1	3	2	4	5

PROTEINS						
		δ				
	Original	**3**	**3.25**	**3.5**	**3.75**	**4**
SAGPool	72.68	73.04	73.84	73.57	71.60	71.79
GMT	70.63	71.07	69.46	71.25	70.36	70.27
DiffPool	71.10	67.71	68.07	69.82	69.45	68.72
DMon	75.45	75.09	75.8	75.36	75.27	75.09
MinCut	73.75	72.86	73.57	73.84	73.75	73.48
AdamGNN	78.12	81.77	79.43	79.17	78.91	78.39
HGP-SL	74.64	72.68	73.66	73.30	74.28	72.14
GIN-ϵ	73.12	71.88	71.88	71.07	73.39	70.57
GIN-0	73.42	73.42	73.84	73.12	72.78	72.50
GCN	65.18	66.07	63.39	66.07	64.28	63.39
Mean	72.81	72.56	72.29	72.66	72.41	71.63
Rank	1	3	5	2	4	6

	NCI1			**NCI109**		
		δ			δ	
	Org.	**2.5**	**3**	**Org.**	**2.5**	**3**
SAGPool	70.10	68.49	70.72	67.37	66.84	67.07
GMT	60.29	57.42	58.44	48.86	51.01	51.13
DiffPool	66.96	67.64	68.18	67.60	65.97	67.2
DMon	74.20	74.23	74.01	72.61	73.50	73.38
MinCut	81.85	82.46	82.50	73.28	73.04	72.56
AdamGNN	81.70	81.21	81.87	65.16	61.79	67.81
HGP-SL	71.53	73.48	70.07	72.87	72.58	72.56
GIN-ϵ	81.85	82.46	82.50	75.93	75.98	76.61
GIN-0	81.70	81.21	81.87	74.99	74.53	74.94
GCN	71.53	73.48	70.07	75.60	76.81	72.70
Mean	74.17	74.21	74.02	69.43	69.21	69.60
Rank	2	1	3	2	3	1

TABLE VI: Training Parameters in Models (Part 1)

model	**lr. rate**	**# epochs**	**# layers**	**hid. size**
SAGPool	0.005	1,000,000	3	128
GMT	0.0005	500	3	32
DiffPool	0.001	500	3	64
DMon	0.001	500	3	32
MinCut	0.0005	15,000	3	32
AdamGNN	0.01	200	1 or 2	64
HGP-SL	0.001	1,000	3	128
GIN-(ϵ & 0)	0.01	350	5	16
GCN	0.005	350	4	128

B. Parameter Details

Table VI and VII represent all the baseline models' parameters' details. A notable observation is that the number of maximum epochs for the SAGPool and MinCutPool seems endless. However, due to the patience variable, models take a much smaller number of epochs during the experiment. The

TABLE VII: Training Parameters in Models (Part 2)

model	weight dec.	patience	batch Size	dropout
SAGPool	0.0001	100	128	50%
GMT	0.0001	50	128	50%
DiffPool	Default	50	128	No
DMon	Default	50	128	No
MinCut	0.0001	50	128	No
AdamGNN	Default	50	64	50%
HGP-SL	0.001	50	512	No
GIN-(ϵ & 0)	0.5	No	128	50%
GCN	0.0001	100	128	50%

AdamGNN model determines the number of layers for the experiment by analyzing the graphs' structural properties. All the models are developed in the PyTorch library and utilize the Adam optimizer where the default weight decay is set to 0 in most cases. Except for AdamGNN (64) and HGP-SL (512), the batch size is 128 for all of the other models. All the baseline models employ different learning rates for evaluation. Six of the models employ the dropout parameter with a rate of 50%, while the other four models abstain from utilizing it.

HeTAN: Heterogeneous Graph Triplet Attention Network for Drug Repurposing

Farhan Tanvir
Department of Computer Science
Georgia State University
Atlanta, Georgia, USA
ftanvir@gsu.edu

Khaled Mohammed Saifuddin
Department of Computer Science
Georgia State University
Atlanta, Georgia, USA
ksaifuddin1@student.gsu.edu

Tanvir Hossain
Department of Computer Science
Georgia State University
Atlanta, Georgia, USA
thossain5@student.gsu.edu

Arunkumar Bagavathi
Department of Computer Science
Oklahoma State University
Stillwater, Oklahoma, USA
abagava@okstate.edu

Esra Akbas
Department of Computer Science
Georgia State University
Atlanta, Georgia, USA
eakbas1@gsu.edu

2024 IEEE 11th International Conference on Data Science and Advanced Analytics (DSAA) | 979-8-3503-6494-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/DSAA61799.2024.10722832

***Abstract*—Modeling the interactions between drugs, targets, and diseases has significant implications for drug discovery, precision medicine and personalized treatments. Current computational approaches consider pairwise interaction, including drug-target or drug-disease interaction individually. On the other hand, within human metabolic systems, the interaction of drugs with protein targets in cells influences target activities. Moving beyond binary relationships and exploring tighter relationships together as triple is essential to understanding drugs' mechanism of action (MoAs). Moreover, considering the heterogeneity of drugs, targets, and diseases, along with their distinct characteristics, it is critical to model these complex interactions appropriately. To address these challenges, we develop a novel Heterogeneous Graph Triplet Attention Network (`HeTAN`) by modeling the interconnectedness of all entities in a heterogeneous graph. `HeTAN` introduces a novel triplet message passing and triplet-wise attention mechanism within this heterogeneous graph structure. In contrast to focusing only on pairwise attention as the importance of an entity for the other, we define triplet attention to model the importance of pairs for the other in the drug-target-disease triplet prediction problem. We perform extensive experiments on real-world datasets and our results show that `HeTAN` outperforms several baselines, demonstrating its superior performance in uncovering novel drug-target-disease relationships.**

***Index Terms*—Drug Discovery, Heterogeneous Graph Neural Network, Graph Neural Network, Representation Learning, Graph Attention, Triplet Prediction**

I. Introduction

Understanding drugs' mechanism of action (MoA) is crucial for drug repurposing, a promising approach to accelerating drug discovery and offering avenues for personalized medicine and targeted therapies. However, traditional drug discovery is time-consuming and expensive [1]. To address this challenge, computational methods have emerged as invaluable tools for leveraging large-scale chemical and genomic data [2].

Recent machine learning advancements have enhanced the study of drugs' MoAs through various learning tasks like drug behavior analysis, target activity evaluation, and disease modeling. [3]. Among these tasks, predicting the relations of drugs with other entities, such as drug-disease and drug-target prediction, have gained significant attention [4], [5]. While existing methods have made progress in predicting the relations of drugs with other entities, they often treat these tasks as isolated tasks, leading to limitations in capturing the interconnected nature of drugs with other entities. Crucially, a drug's therapeutic effect hinges on its interplay with biological targets within complex pathways and the overall metabolic system [1]. Drugs interact with protein targets in cells to modulate target activities, altering biological pathways to treat diseases. This activity integrates higher-order relationships among multiple entities. Therefore, a more comprehensive triple relationship involving drugs, targets, and diseases must be considered to capture the interplay between these entities.

Tensor factorization has emerged as a popular approach for drug-target-disease triplet prediction problems. They infer missing entries in drug-target-disease tensors via extracting latent structures from high-dimensional data [6]. NeurTN [7] combines tensor algebra and deep neural networks to learn the intrinsic relationships among drugs, targets, and diseases. However, traditional tensor models like Canonical Polyadic (CP) decomposition and Tucker decomposition suffer from issues, including linearity and data sparsity. Nonlinear tensor factorization methods have shown promise in capturing the complexities of the data, but they often rely on prior Gaussian processes that are challenging to estimate [8]. Moreover, incorporating auxiliary information into tensor models requires tedious feature engineering, making it challenging to handle large-scale healthcare data [9]. Furthermore, while many graph-based machine-learning models are common for drug-related problems, the tensor model does not utilize graph machine-learning models to predict new triplets.

Heterogeneous graphs, also called Heterogeneous Information Networks (HIN) [10], provide a robust framework for representing diverse entities and interactions in drug discovery.

In these graphs, nodes represent entities like drugs, proteins, pathways, chemical substructures, ATC codes, and diseases, while edges capture interactions between them. While many models are developed to represent the relationship between drug, target, and disease, they focus on predicting pairwise relations between drug and other entities such as drug-drug, drug-disease and drug-target [11]–[15]. These methods base their predictions on established drug-drug similarity, target-target similarity as well as known drug-target associations. However, there is no HIN-based triplet prediction model.

To address these limitations and model the complex interactions between drugs, targets, and diseases more effectively, we propose a novel **He**terogeneous Graph **T**riplet **A**ttention **N**etwork (HeTAN). HeTAN leverages the power of heterogeneous graphs, representing diverse entities and their interactions, and employs a novel triplet attention mechanism to capture higher-order interactions within the drug-target-disease triplets. We capture higher-order interactions between drug, target, and disease through a triplet-wise attention mechanism. This gives us a more comprehensive understanding of drug MoAs and can accelerate drug repurposing for personalized medicine. While it is defined for drugs, targets, and diseases triplets, it is a generic model that can be applied to other triplets. Our main contributions are as follows:

- **Utilizing heterogeneous graph neural network for drug-target-disease triplet prediction:** We propose a novel approach that models the complex interactions between drugs, targets, and diseases using a heterogeneous graph neural network (HGNN). By incorporating different types of nodes and edges, our approach effectively captures the rich information embedded in the interactions between these entities, leading to improved prediction performance.
- **Introducing the HeTAN model:** We develop a novel model, HeTAN, by proposing a novel triplet message passing and triplet-wise attention mechanisms on different types of entities in a heterogeneous graph. Our model goes beyond the pair-wise interaction and captures higher-order triplet-wise interactions to make triplet predictions on the heterogeneous graph. While triplet message passing enables passing the information among three different entities (drug-target-disease), the triplet attention mechanism enables the model to focus on the most relevant pairs for an entity instead of the most relevant neighbor. These enhance its predictive accuracy and its ability to capture intrinsic and complex interactions among three entities. No prior work in GNN and HGNN has explored triplet-wise message-passing and attention mechanisms.
- **Extensive Experiments:** We conduct extensive experiments to show the effectiveness of our model on two different datasets. We also compare the proposed HeTAN model with several baseline models. The results with different accuracy measures show that our method significantly surpasses the baseline models. In addition, different case studies denote that different datasets and external literature evidence can validate our model's predictions.

The remainder of this paper is organized as follows: Section II reviews related works. Section III describes the creation of a heterogeneous graph. Details of the HeTAN model are presented in Section III. Experiments and results are discussed in Section IV. Finally, Section V concludes the paper.

II. Related Works

This section provides an overview of existing research in computational predictions of drugs, targets, and diseases, specifically on triplet prediction for high-dimensional structured data.

A. Modeling drug-target-disease

Treating human diseases involves interactions among drugs, biological targets, and disease pathways. Computational pharmacology seeks to uncover associations among these entities and understand drugs' mechanisms of action (MoAs) [1]. A common technique involves network-based inference models, such as bipartite networks with distinct layers for drugs and diseases (targets). Various machine learning methods, including random walks, matrix factorization, and support vector machines [16], have been used to predict new drug-disease and drug-target interactions.

DTINet [15] integrates diverse drug-related information to build a heterogeneous network and employs a compact feature learning algorithm to derive low-dimensional vector representations of nodes. This model uses a known set of drug-target associations as a reference to determine the optimal projection from the drug space onto the protein space, ensuring that the projected feature vectors of drugs closely align with the feature vectors of their known targets. Chen et al. [16] integrated a protein-protein similarity network, a drug-drug similarity network, and a drug-target interaction network into a heterogeneous network. Using a random walk algorithm, they inferred new drug-target connections without directly modeling drug-disease relationships, focusing on predicting drug-target interactions by learning a transformation matrix from known interactions. Similarly, Fu et al. [17] utilized known drug-target connections from various data sources but did not explicitly use drug-disease-target triples, thus only predicting drug-target interactions. Zheng et al. [18] developed a matrix factorization method based on the similarity of chemical structures and protein sequences to establish drug-target relationships. These methods rely on chemical structure similarity through structural fingerprints and protein sequence-based similarity. However, these approaches treat drug-disease and drug-target predictions as separate tasks, limiting a comprehensive understanding of the interconnected drug-target-disease relationships.

B. Triplet Prediction

Triplet prediction has broad applications, ranging from drug repurposing to natural language processing and computer vision. Zhang et al. [19] introduced an attention mechanism based on transformers to capture relationships between three

entities (query, key, and value) for improved language understanding and generation. In natural language processing, triplet prediction has been used for tasks like relationship extraction, where models use sentence-level attention and entity descriptions to predict relationships between entity triplets in text [20]. In computer vision, triplet prediction techniques have been used for face recognition and person re-identification [21].

In drug-target-disease prediction, various models have been developed, such as collective matrix factorization [22] and neural tensor networks [7], to capture nonlinear dependencies within triplets. Recent research explores the interdependence of drugs, targets, and diseases through event-graph modeling and neural tensor network models [7], [23]. Meanwhile, recent advancements have explored the potential of hypergraphs and hypergraph neural networks in biomedical problems [24], [25], including drug-microbe-disease associations, [26]. A hypergraph is a unique graph with hyperedges. Unlike a regular graph where the degree of each edge is 2, hyperedge is degree-free; it can connect an arbitrary number of nodes. While these approaches are significant, the application of heterogeneous graph neural networks (HGNNs) and triplet attention mechanisms remains largely unexplored.

Despite many proposed models for triplet prediction, most focus on homogeneous entities, and none have applied graph-based models to drug-related problems. `HeTAN` distinguishes itself in drug repurposing by pioneering the combined use of HGNNs and triplet attention mechanisms. While defined for drug-target-disease triplets, `HeTAN` can be applied to other triplets as well.

III. Methodology

Given that a triplet includes a drug, a target, and a disease, our goal is to predict whether the triplet has an interaction. In this paper, we propose a novel approach that leverages the power of heterogeneous information networks (HIN) and introduces the concept of triplet attention. To achieve this, we develop the Heterogeneous Graph Triplet Attention Network (`HeTAN`), which employs an end-to-end encoder-decoder architecture. The encoder integrates a triplet attention mechanism to determine the significance of pairs (e.g., target-disease) for the other entity (e.g., drug) while learning embeddings of all entities and triplets. Moreover, `HeTAN` incorporates a decoder that learns and predicts the interaction between entities of triplets. The system architecture of `HeTAN` is outlined in Figure 1. We optimize the model parameters with a cross-entropy loss function.

Our proposed model consists of the following steps:

1) Heterogeneous Graph Construction & Node's Feature Extraction
2) Heterogeneous Graph Triplet Attention Network Architecture
   - Encoder: Triplet Attention-based node representation learning
   - Decoder: Drug-Target-Disease triplet prediction

A. Heterogeneous Graph Construction & Node's Feature Extraction

The first step in our approach is to construct a heterogeneous graph that captures the complex relationships among drugs, proteins (targets), and diseases. The graph consists of three types of nodes: *drugs*, *proteins*, and *diseases*. We establish edges between these nodes based on known drug-target interactions and drug-disease associations. This construction allows us to represent the rich interactions and dependencies between different entities in the graph.

Graph neural network (GNN) models can optimize and refine node representations with an iterative learning process. These models transform initial node attributes or features with message passing and aggregation mechanisms from the node's neighbors to generate enriched and effective node vector representations. In our next step, we utilize the structural properties of drugs and targets to extract node features from our heterogeneous graph. Specifically, we focus on the chemical substructures of drugs and targets, represented as SMILES strings [27] and Amino Acid sequences, respectively. We employ the Explainable Substructure Partition Fingerprint (ESPF) [28] algorithm to create drug and target features. ESPF decomposes the SMILES string and Amino Acid sequence into frequent substructures, selecting the most significant ones based on a frequency threshold. These substructures provide informative features for the drugs and targets utilized in the subsequent steps of the `HeTAN` model. We represent disease nodes with one-hot encoded representations. After constructing the heterogeneous graph and extracting node features, we propose the `HeTAN` architecture for learning representations that capture complex relationships among drugs, targets, and diseases.

B. Heterogeneous Graph Triplet Attention Network

The core goal of our research is to address the challenge of predicting drug-target-disease interactions. To achieve this, our model `HeTAN` leverages the rich information in heterogeneous networks and captures the complex relationships among drugs, targets, and diseases. The model is trained using an end-to-end approach to predict drug-target-disease interactions. The model is responsible for aggregating information from neighboring nodes and learning higher-order relationships in the graph using a triplet attention mechanism, the critical component of our model. The triplet attention mechanism calculates attention coefficients based on the features of all three nodes in a triplet (i, j, and k), where i is the central node, and j and k are neighboring nodes. When aggregating information, these coefficients are used to weigh the importance of neighboring node pairs for the central nodes in each triplet. The attention mechanism is applied to all graph-generated triplets, enabling the model to capture complex, interconnected relationships among the different node types. Afterward, the encoded representations of nodes are obtained by aggregating information from neighboring nodes, weighted by the attention coefficients. This process is repeated for each layer of

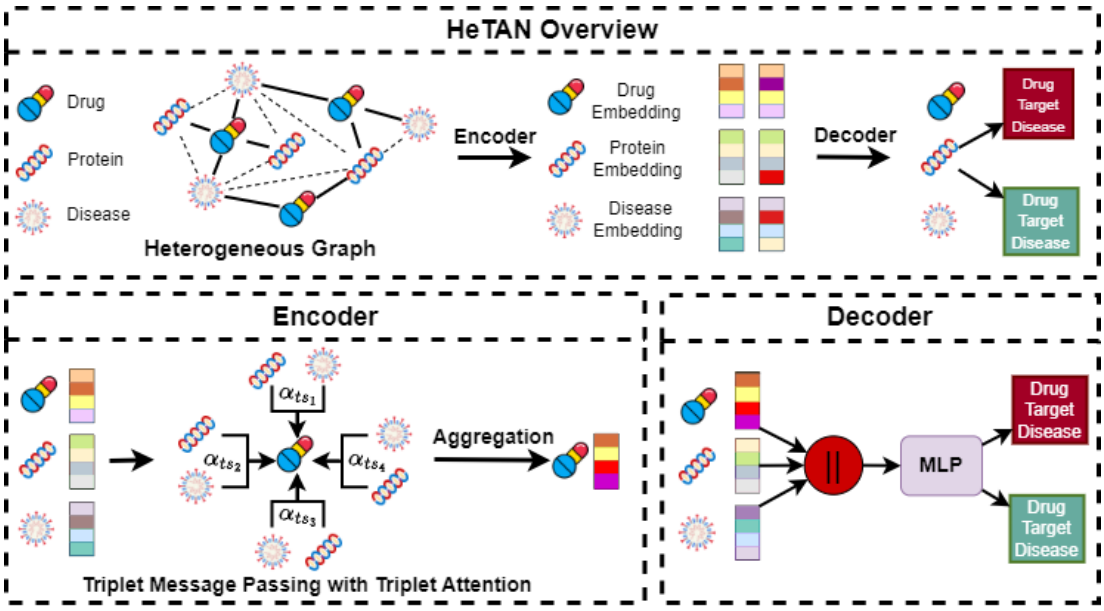


Fig. 1: The HeTAN workflow comprises three steps: Heterogeneous Graph Construction, Encoder, and Decoder. Initially, a heterogeneous network is built with drug, target, and disease nodes connected by drug-target or drug-disease edges. Target-disease connections (dashed lines) are inferred from shared drug associations. We introduce Triplet Message Passing (TMP) and Triplet-wise attention to generate node representations. Finally, using concatenated node representations, a Multi-Layer Perceptron (MLP) predicts drug-target-disease interactions.

the HeTAN model, allowing the model to learn increasingly complex patterns and dependencies across multiple layers.

Our proposed model consists of the following steps:

- Encoder: Triplet Attention-based Node Representation learning
- Decoder: Predicting Drug-Target-Disease Interactions

*1) **Node Representation Learning (Encoder)**:* The encoder component of our model focuses on learning informative node representations. In a heterogeneous graph, nodes and edges belong to different types, and each type of node has its own distinct feature space. To effectively learn informative node representations, we must align these diverse feature spaces into a common one. This enables meaningful comparisons and interactions among nodes of different types. To overcome this challenge, we introduce a type-specific transformation matrix M, which projects the features of different nodes into a common feature space as $h_i^{'} = M \circ h_i$

Traditional Graph Convolutional Networks (GCNs) rely on pairwise message passing, where neighboring nodes pass messages to each other. However, this approach falls short in capturing intricate dependencies beyond pairwise relationships, especially in our case, where understanding drug-target-disease interactions requires considering the complex relations inherent in drug-target-disease triplets.

To solve this limitation, we define the **Triplet Message Passing** function (TMP), a novel mechanism that leverages node triplets for representation learning. Instead of pairwise interactions, **TMP** considers neighboring node pairs and their influence on the central node. For a central node of type i, we define its neighbors as node pairs (N_i) comprising node pairs of type j and k as $N_i = \{(j_1, k_1), ..., (j_n, k_n)\}$. We pass messages from these neighbor pairs to the central node i. This allows the model to capture richer contextual information and complex relationships in drug-target-disease interactions. The triplet message passing function is defined as:

$$z_i^l = TMP(z_i^{l-1}, N_i) \tag{1}$$

For one central node, there are several node pairs as the neighbors. However, it is essential to note that not all neighbor pairs are equally crucial for the central node. Message passing should consider these varying levels of importance. We design a novel **Triplet-wise Attention** mechanism to incorporate the importance of neighbor pairs for a central node into message passing. This attention mechanism utilizes the features of all three nodes in a triplet and assigns attention coefficients, signifying the relative importance of the neighbor pairs for the central node. Based on the features of all three nodes in a triplet, the attention coefficient e_{ijk} is defined as follows:

$$\begin{aligned} e_{ijk} &= a(h_i^{'}, h_j^{'}, h_k^{'}) \\ &= LeakyRELU(NN(h_i^{'} || h_j^{'} || h_k^{'})) \end{aligned} \tag{2}$$

In Eq. 2, a denotes the triplet-wise attention mechanism, and $||$ denotes the concatenation operation. We employ a neural network in the attention mechanism, denoted as NN. This neural network is designed to capture essential relationships and dependencies among the nodes in a triplet. Additionally,

to capture the nonlinear dependencies among drug-target-disease data, we apply the LeakyReLU activation function. LeakyReLU is chosen for its ability to introduce nonlinearity in the model, allowing it to capture complex relationships critical for accurately predicting drug-target-disease interactions.

It is vital to make attention coefficients easily comparable across different nodes. Therefore, the attention coefficients are then normalized using a Softmax function. This step ensures that the model appropriately weighs the attention of each neighbor when aggregating information. The normalized attention coefficient α_{ijk} is defined as follows;

$$\begin{aligned}\alpha_{ijk} &= softmax_j(e_{ijk}) \\ &= \frac{\exp(e_{ijk})}{\sum_{l,m\in N(i)} \exp(e_{ilm})}\end{aligned} \tag{3}$$

During the triplet message passing process, it is imperative to consider the message from neighbor pairs. To generate messages from pairs, we concatenate the representations of the nodes within the pair. We then pass this concatenated feature vector of size $2d$ through a single-layer feedforward neural network to transform it into a feature vector of size d. After multiplying each pair message with calculated pair-wise attention, these messages are aggregated and combined with the central node's representation using a self-attention mechanism. This mechanism considers the node's features and aggregated information, allowing the model to capture its unique influence within the heterogeneous graph. So, the triplet message passing function, **TMP** in Eq 1 is defined as follows:

$$z_i = \delta(h_i^{'} + W \circ \sum_{j,k\in N(i)} (\alpha_{ijk} \circ NN(h_j^{'} || h_k^{'}))) \tag{4}$$

where $\circ$ represents multiplication operator and W is a trainable parameter. To incorporate self-attention, we use W, which determines the weight or importance of node i's embedding in the aggregation process. The purpose of W is to control the influence of node i's features and the aggregated features of its neighboring nodes on the overall representation.

We employ multi-head attention to capture more complex patterns and relationships, each focusing on different aspects of the data. This further enhances the model's ability to learn intricate patterns and relationships within the heterogeneous graph. In multi-head attention, multiple attention mechanisms (K) are used individually to transform the features, and the outputs are concatenated $(||)$ to obtain the final representation. So, the final triplet message passing with multi-head attention is defined as follows;

$$z_i = ||_{k=1}^{K} \delta(h_i^{'} + W \circ \sum_{j,k\in N(i)} (\alpha_{ijk} \circ NN(h_j^{'} || h_k^{'}))) \tag{5}$$

By integrating these equations and steps, our model learns informative node representations within a heterogeneous graph. This ensures it captures complex relationships, intricate patterns, and crucial interactions for predicting drug-target-disease interactions.

*2) **Drug-Target-Disease Triplet Prediction (Decoder)**:* Our model has a decoder component that predicts the likelihood of interactions between drugs, targets, and diseases as new triplets based on the representations of entities obtained from the encoder. Decoder, in particular, assigns a score to drug, target, and disease triplet (v_i,v_j,v_k), expressing how likely it is that drug v_i target v_j, and disease v_k are interacting. The corresponding entities' features are concatenated and passed through a multilayer perceptron (MLP).

$$pred_{x,y,z} = MLP(z_x || z_y || z_z) \tag{6}$$

The MLP outputs a prediction score, Y', between 0 and 1. A score close to 1 indicates a high likelihood of interaction among the triplets, whereas a score close to 0 indicates less likely interaction.

*3) **Model Optimization**:* We train our entire encoder-decoder architecture as a binary classification problem by minimizing a binary cross-entropy loss function specified as follows:

$$L = -\sum_{i=1}^{N} Y_i \log Y_i^{'} + (1 - Y_i) \log(1 - Y_i^{'}) \tag{7}$$

where N is the total number of triplets, Y_i is the actual label indicating the presence or absence of an interaction for the triplet, and $Y_i^{'}$ is the predicted score for the triplet.

C. Analysis of HeTAN model

Here we give the analysis of HeTAN as follows:

- **Handling Diverse Nodes and Relationships:** HeTAN effectively handles different types of nodes and relationships, integrating rich semantics within a heterogeneous graph. We facilitate message passing among neighbor node pairs and a given node. During this message passing, we incorporate the importance of neighbor pairs for a central node through a novel Triplet-wise Attention mechanism. Leveraging Triplet Message Passing and Triplet-wise Attention allows for enhanced integration, promotion, and improvement of diverse node embeddings.
- **Efficiency and Complexity:** The proposed HeTAN is highly efficient and can be easily parallelized. The complexity of HeTAN can be analyzed based on its main components: heterogeneous graph construction, triplet attention mechanism, and message passing. The initial step of constructing a heterogeneous graph involves nodes (drugs, targets, diseases) and edges (interactions), with a complexity of $O(|V| + |E|)$. For each triplet of nodes (i, j, k), the triplet attention mechanism computes attention scores and normalizes them, resulting in a complexity of $O(|E|ff' + |V|df')$, where f is the initial feature dimension, f' is the output feature dimension, and d is the average degree of nodes.

  HeTAN leverages multi-head attention to capture complex patterns, scaling the computation by the number

TABLE I: Statistics of Dataset

# of Instances	DrugBank (DB)	DrugBank and CTD
Drugs	531	450
Targets	836	708
Diseases	279	1, 267
Triplets	27,238	175, 288

TABLE II: Hyper-parameter Settings

Parameter	Values
Learning rate	1e-2, 5e-2, 1e-3, 5e-3, 1e-5
Number of heads per layer	8, 16, 32
Hidden units	8, 16, 32, 64, 128
Dropout	0.1, 0.3, 0.5, 0.6
Weight decay	0.01, 0.001

of heads K. The message passing and aggregation process, combined with multi-head attention, contributes to an overall complexity of $O(K(|E|ff' + |V|df'))$. This efficient handling of heterogeneous graphs and higher-order interactions enables HeTAN to effectively capture intricate relationships among drugs, targets, and diseases, demonstrating its capability in drug-target-disease triplet prediction.

IV. Experiment

To evaluate our HeTAN model, we conduct experiments involving negative sampling and random dataset splitting into train and test sets. Our performance assessment include Recall, Precision, F1-score, AUROC, and the commonly used top-n metric hit@n. This section summarizes our experimental parameters, evaluation protocols, and analysis of results.

A. Datasets, Parameter Settings & Baselines

Our study utilizes data from DrugBank and CTD, providing insights into drug-related information. Two dataset configurations are employed. One uses data exclusively from DrugBank, encompassing details about drug-target interactions and drug-disease associations. The other configuration integrates information from DrugBank (concerning drug-target interactions) with data from CTD (providing drug-disease associations). This integration provides a comprehensive view of <drug, target, disease> triplets. Subsequently, the DrugBank and combined datasets will be referred to as DB and DB&C, respectively. Table I summarizes the vital characteristics of nodes and edges in the heterogeneous graph.

Datasets are split into random training (80%) and testing (20%) subsets for five iterations. This splitting process is repeated five times, and the average accuracy metrics are calculated and reported in the results section. The optimal hyper-parameters are obtained by grid search based on the validation set. The ranges of grid search are shown in Table II. We train the HeTAN model using the cross-entropy loss and optimize the model parameters using the Adam optimizer. The optimal learning rate is determined to be 1e-5, and the optimal dropout rate is found to be 0.6 to prevent overfitting. Training runs for 2000 epochs with early stopping after 200 consecutive epochs without validation loss improvement.

To construct negative triplets, we employ negative sampling by randomly replacing one or all nodes in positive triplets, ensuring they are absent from the actual data. We assess HeTAN's performance through a diverse set of metrics encompassing accuracy, precision, F1-score, and AUROC. We also utilize the commonly used top-n metric hit@n and NDCG@n, as proposed by [29], [30]. Hit@n measures whether a test triplet appears within the top-n ranked predictions, while NDCG@n prioritizes higher-ranked matches. We rank triplets in descending order based on model prediction scores, prioritizing those most likely to represent valid interactions.

To evaluate HeTAN's effectiveness, we compare it to a range of state-of-the-art models categorized by their approaches:

- *Tensor Decomposition Methods:* CP and Tucker are famous tensor models with diverse variants that are being successfully applied in health data analysis [31]. They both adopt multilinear assumptions.
- *Attention-based Methods:* We use transformer, a robust deep learning architecture that captures complex relationships and patterns in the data. It utilizes self-attention mechanisms to effectively learn and represent the interactions between drugs, targets, and diseases. For these models, the embeddings of the triplet nodes (drug, target, disease) are concatenated, and the combined embeddings are used to predict interactions.
- *Graph Neural Network (GNN):* We use GNN architectures on our heterogeneous graph to learn the representation of nodes. We select three standard GNN-based methods: GIN [32], GAT [33], and GraphSAGE [34]. Among these GNN models, GAT [33] uses pairwise attention to generating node representation. Similar to attention-based methods, GNN models concatenate the embeddings of the triplet nodes and use these concatenated embeddings for interaction prediction.
- *Heterogeneous Graph Neural Network (HGNN):* For this baseline, we use the commonly used HGNN model heterogeneous graph transformer (HGT) [35]. HGT incorporates pairwise attention on a heterogeneous graph to learn the representation of nodes. The triplet node embeddings are concatenated and used to predict interactions.
- *NeurTN:* Neural Tensor Network (NeurTN) [7] combines tensor algebra and deep neural networks, offering a more powerful way to capture the nonlinear relationships among drugs, targets, and diseases. Both NeurTN and HeTAN combine drug-target and drug-disease interactions from DrugBank and CTD.
- *DDTE:* Moon et al. [36] construct a heterogeneous knowledge graph including various drug-related information and utilize TransE [30] model to infer drug–disease–target relationships.
- *MHGNN:* MHGNN-DTI [37] builds the model with a dual-channel architecture to learn drug and target embeddings, respectively, using a graph attention mechanism and metapath techniques. It proposes building correlation graphs to exploit high-order relations. Finally, it performs

TABLE III: Comparing performance of HeTAN with other baseline models on DB

Model	Method	F-1 Score	Precision	Recall	ROC-AUC	AUPR
Tensor-based	TD	47.00	48.51	45.59	49.19	48.98
	CPD	52.91	52.19	56.19	49.84	50.06
Attention-based	Transformer	52.31	62.96	51.18	60.62	59.46
GNN-based	GraphSAGE	72.24	61.3	83.94	66.4	59.92
	GIN	74.06	71.18	77.2	73.08	66.31
	GAT	72.63	62.65	82.34	67.64	60.92
HGNN-based	HGT	80.44	82.71	79.41	83.13	83.32
	MHGNN	81.9	87.16	83.86	92.63	91.57
	HeTAN	**86.31**	**88.43**	**84.34**	**93.46**	**93.07**

TABLE IV: Comparing performance of HeTAN with other baseline models on DB&C

Model	Method	F-1 Score	Precision	Recall	ROC-AUC	AUPR
Tensor-based	TD	53.17	62.45	47.86	60.65	62.04
	CPD	57.23	63.72	52.19	59.82	60.76
Attention-based	Transformer	83.05	85.24	81.09	82.04	75.36
GNN-based	GraphSAGE	83.31	78.97	90.04	75.98	70.72
	GIN	83.98	78.16	83.49	76.45	71.52
	GAT	85.17	82.44	82.11	83.76	77.15
HGNN-based	HGT	85.22	87.08	84.55	87.98	84.07
	MHGNN	87.7	85.88	88.79	95.64	94.45
	HeTAN	**90.91**	**93.12**	**89.88**	**98.01**	**97.75**

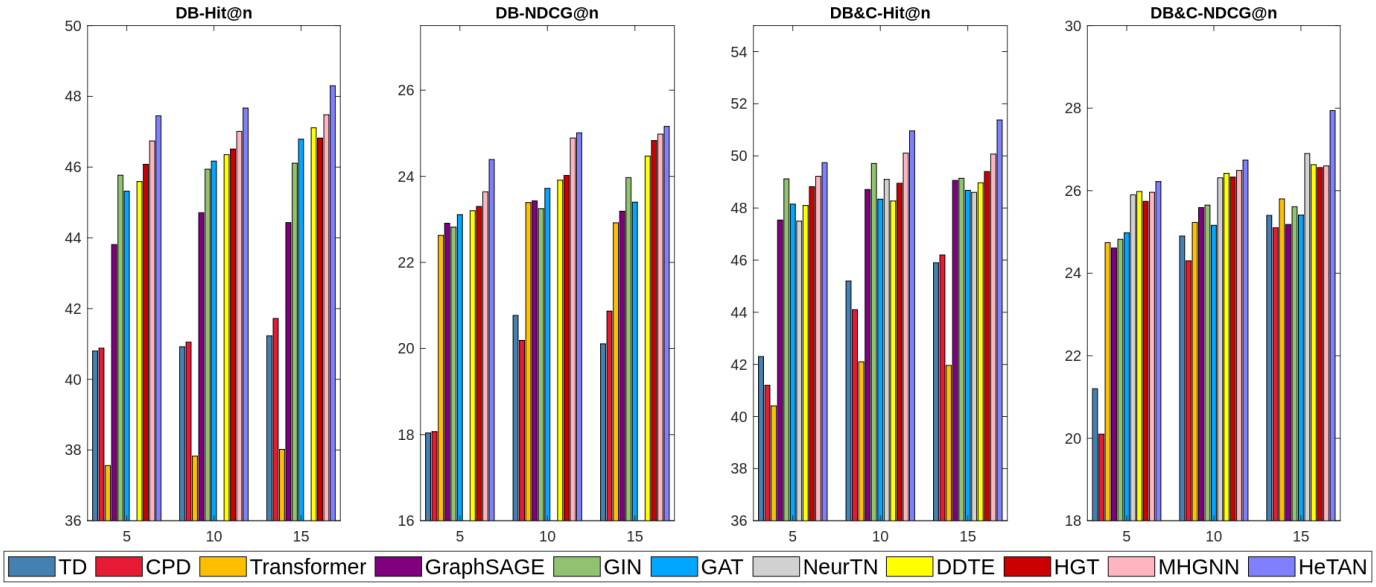


Fig. 2: Evaluation of top-n performance for HeTAN and other baseline models in terms of a) Hit@n and b) NDCG@n on DB and DB&C

pairwise drug-target interaction prediction.

B. Comparison with baselines

In this study, we conduct a comprehensive performance analysis of HeTAN compared to a selection of state-of-the-art baseline models. We employ diverse performance metrics to assess these models' efficacy. Specifically, we report the F-1 Score, Precision, Recall, ROC-AUC, and AUPR results in Table III for DB and Table IV for DB&C. Both tables refer to the tensor-based baselines, tucker decomposition and CP decomposition as TD and CPD, respectively. Our model, HeTAN, outperforms all baseline models for both datasets, showcasing its exceptional predictive capabilities.

For instance, on DB, HeTAN achieves impressive F-1 score, ROC-AUC, and AUPR of 86.31%, 93.46%, and 93.07%, representing significant improvements over the best-performing baseline, MHGNN, which achieves F-1 score, ROC-AUC, and AUPR of 81.9%, 92.63%, and 91.57%, respectively. The superior performance of HeTAN is further evident in the DB&C dataset, where it attains F-1 score, ROC-AUC, and AUPR of 90.91%, 98.01%, and 97.75%, surpassing the performance of other models by a considerable margin.

TABLE V: Novel Triplet Predictions by HeTAN FROM DB&C

Drug	Target	Disease	DB&C Label	Prediction	DB Label
Carbamazepine	NR1I2-HUMAN	Osteoporosis	0	0.99	1
Testosterone	ERR3-RAT	Myocardial infarction	0	0.98	1
Nefazodone	DRD2-HUMAN	Schizophrenia	0	0.97	1
Raloxifene	ERR3-RAT	Obesity	0	0.93	1
Fenofibrate	MMP19-HUMAN	Psoriatic arthritis	0	7e-09	0

TABLE VI: Novel Triplet Predictions by HeTAN FROM DB

Drug	Target	Disease	DB Label	Prediction	DB&C Label
Cyclobenzaprine	5HT2C-HUMAN	Muscle Spasm	0	0.99	1
Cyclobenzaprine	AA2AR-HUMAN	Gout	0	0.98	1
Imipramine	ADA1D-HUMAN	Interstitial Lung Disease	0	0.97	1
Quetiapine	HRH1-HUMAN	Schizophrenia	0	9.9e-10	0
Verapamil	CAC1S-HUMAN	Cluster headache	0	5e-07	0

TABLE VII: Top five drug-target pairs predicted by our proposed HeTAN for depression

Drug (DrugBank)	Target (UniProt)	Evidence
Amitriptyline	Sodium-dependent serotonin transporter	Kim Lawson [38]
Nortriptyline	5-hydroxytryptamine receptor 2A	Pierre Blier [39]
Imipramine	Sodium-dependent serotonin transporter	Dempsey et al. [40]
Nortriptyline	Muscarinic acetylcholine receptor M5	Philip et al. [41]
Nortriptyline	MD(2) dopamine receptor	Pierre Blier [39]

In addition to these performance metrics, we adopt the top-n metrics, Hit@n and NDCG@n, as illustrated in Figure 2. These metrics are particularly critical in triplet prediction, as they assess the ranking quality of the model's predictions. HeTAN's top-n metrics (Hit@n and NDCG@n) performance showcases its superior ranking ability, which is crucial for accurate triplet prediction. On DB, HeTAN achieves a Hit@15 score of 50.11% and NDCG@15 of 27.36%, significantly exceeding the top baseline (MHGNN) by over 6% and 3%, respectively.

Tensor-based models exhibit solid performance but often lag in recall and ranking. Attention-based methods, like the Transformer, improve Precision and Recall. NeurTN, combining tensor and attention models, excels in top-n metrics. Since different accuracy results like F-1 score, Precision and Recall are unavailable on NeurTN paper, we could not present and analyze these results with other baseline models. Similarly, we could not obtain these results from DDTE.

GNN and HGNN-based models consistently achieve F-1 scores surpassing 70%, emphasizing the pivotal role of graph structural information. GNN and HGNN-based models represent interactions between graph nodes and capture graph dependence through message passing. Comparing graph attention-based models, GAT and HGT rely on pairwise attention, and HeTAN utilizes triplet-wise attention. HeTAN consistently performs better than GAT and HGT. For example, GAT achieved F-1 scores of 72.63% on DB and 85.17% on DB&C, while HGT scored 80.44% on DB and 85.21% on DB&C. One notable heterogeneous graph neural network, MHGNN achieves F-1 scores of 81.9% on DB and 87.7% on DB&C, demonstrating strong performance. MHGNN's strength lies in its dual-channel architecture and meta-path techniques to exploit high-order relations. Still, MHGNN falls short of HeTAN's results. HeTAN achieve 86.31% and 90.91% on DB and DB&C, respectively.

HeTAN effectively manages diverse nodes and relationships, integrating rich semantics within a heterogeneous graph. By using triplet message passing and triplet-wise attention, the model captures intricate patterns and dependencies, offering a comprehensive understanding of drug-target-disease associations. Multi-head attention enhances its ability to learn from complex data, ensuring robust predictions. Overall, HeTAN significantly improves prediction accuracy, positioning itself as a powerful tool for drug discovery and personalized medicine.

C. Prediction and Validation of Triplets

This study evaluates HeTAN's ability to predict drug-target-disease interactions using real-world datasets. To determine its effectiveness in predicting missing interactions, we compare HeTAN's predictions with data from two distinct datasets.

We start by selecting triplets from Dataset DB&C, which lack interaction data in DB&C but possess relevant association information in DB. We train HeTAN on the DB&C dataset, ensuring that the selected triplets are used exclusively in the test set to minimize potential bias. The predicted scores for these triplets, presented in Table V, consistently exceed 90%, suggesting that these triplets are likely to exhibit interactions despite the absence of explicit interaction data in DB&C. To validate these predictions further, we cross-reference them with the information in DB. Remarkably, this comparison confirms the interactions between these triplets, reinforcing the predictive power and accuracy of HeTAN.

To expand our validation process and test HeTAN's generalizability, we select another set of five drug triplets from DB, which lack interaction information within DB but contain such data in DB&C, as highlighted in Table VI. We train HeTAN using the DB dataset for this validation, tailoring the model specifically to this unique dataset configuration. Subsequently, we validate the predicted scores by cross-referencing them with DB&C, which serves as an independent

validation set. Validating predicted scores against DB&C emphasizes HeTAN's reliability and generalizability, showcasing its adaptability across datasets and reinforcing its real-world predictive capabilities.

D. Case Study on Depression

Personalized treatment is a core objective in our medical research, particularly in identifying effective drugs for specific diseases and understanding their biological targets. HeTAN has been employed to uncover new drug-target combinations relevant to depression—a complex condition with various molecular factors. By focusing on triplets where the disease is depression, HeTAN was trained on a heterogeneous graph from DrugBank and CTD datasets. For this experiment, we filter our predicted triplets to focus on those where the disease is depression. Table VII enlists the top five pairs of (drug, target) corresponding to depression and literature evidence supporting these predictions. For depression, these pairs are the highest-ranked predictions based on the model's scoring and have corresponding evidence in the literature, demonstrating their potential relevance and validity. These results underline HeTAN's potential in identifying clinically relevant drug-target pairs, marking a significant step toward personalized medicine. The model's reliable predictions offer a promising approach to revolutionizing treatments for complex diseases like depression.

E. Ablation Study

To assess the contribution of each component in HeTAN, we perform an ablation study with five variants:

- `HeTAN-Sum (HeTAN-S)`: This variant employs summation instead of concatenation and neural network transformations for neighbor embedding in Eq 4.
- `HeTAN-Concat (HeTAN-C)`: Three neighbor node embeddings are concatenated in Eq 2 and then reduced in dimension.
- `HeTAN-Elem-Prod (HeTAN-EP)`: This variant uses the element-wise product on neighbor node embeddings in Eq 4.
- `HeTAN-Triplet-Attention-Sum (HeTAN-TAS)`: In this variant, three neighbor node embeddings are summed in Eq 2 to get triplet-wise attention.
- `HeTAN-Triplet-Attention-Elem-Prod (HeTAN-TAEP)`: This variant applies the element-wise product on three neighbor node embeddings in Eq 2 to get triplet-wise attention.

In comparing the model variants with the original HeTAN, HeTAN-Sum and HeTAN-Concat demonstrate weaker performance, likely due to their use of summation or concatenation, which may not capture complex relationships as effectively as the original approach. Similarly, HeTAN-TAEP and HeTAN-TAS underperform compared to HeTAN, highlighting the efficacy of applying a neural network for concatenated embeddings. As shown in Figure 3, the original HeTAN model consistently surpasses its variants across key metrics like F1-score, Recall, and ROC-AUC, underscoring the effectiveness of its triplet-wise attention and message passing mechanisms. This analysis confirms HeTAN's robustness in managing complex, heterogeneous data in biomedical research.

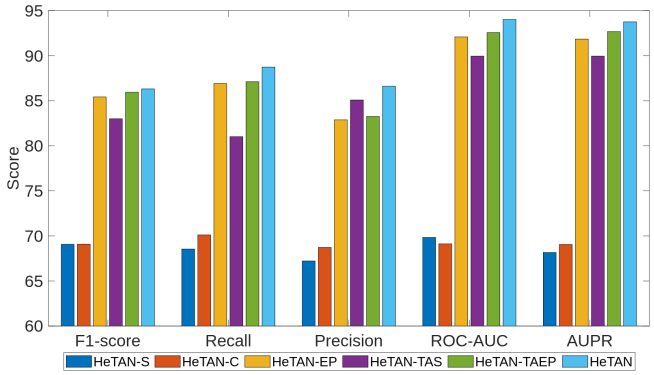


Fig. 3: Performance Comparison of HeTAN with its variants

V. Conclusion

HeTAN stands out as a powerful model for modeling drug-target-disease interactions thanks to its dedicated HGNN architecture and innovative triplet-attention mechanism. This approach effectively addresses limitations encountered in previous models, leading to significant improvements in performance. The novel triplet-attention mechanism holds broad potential for application beyond drug discovery, extending to diverse domains involving heterogeneous graphs and higher-order interactions.

While HeTAN is currently defined for drug-target-disease triplets, future research could further enhance its capabilities by applying it to different triplet combinations and incorporating additional elements, such as drug-target-pathway-disease interactions. This expansion could lead to a deeper understanding of drug mechanisms and improved predictive accuracy. Moreover, integrating multi-omics data and exploring more complex graph structures are promising avenues for boosting HeTAN's predictive power and providing a more comprehensive view of biological processes. These advancements can significantly contribute to progress in personalized medicine and drug development, ultimately benefiting patient outcomes and healthcare systems.

References

[1] A. S. Hauser, M. M. Attwood, M. Rask-Andersen, H. B. Schiöth, and D. E. Gloriam, "Trends in gpcr drug discovery: new agents, targets and indications," *Nature Reviews Drug Discovery*, vol. 16, pp. 829–842, 2017.

[2] H. Chen and J. Li, "Drugcom: Synergistic discovery of drug combinations using tensor decomposition," *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 899–904, 2018.

[3] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinform.*, vol. 34, no. 13, pp. i457–i466, 2018. [Online]. Available: https://doi.org/10.1093/bioinformatics/bty294

[4] A. Ezzat, P. Zhao, M. Wu, X. Li, and C. K. Kwoh, "Drug-target interaction prediction with graph regularized matrix factorization," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, pp. 646–656, 2017.

[5] X.-Y. Yan, S. Zhang, and C. He, "Prediction of drug-target interaction by integrating diverse heterogeneous information source with multiple kernel learning and clustering methods," *Computational biology and chemistry*, vol. 78, pp. 460–467, 2019.

[6] H. Chen and J. Li, "Modeling relational drug-target-disease interactions via tensor factorization with multiple web sources," *The World Wide Web Conference*, 2019.

[7] ——, "Learning data-driven drug-target-disease interaction via neural tensor network," in *International Joint Conference on Artificial Intelligence*, 2020.

[8] S. Zhe, K. Zhang, P. Wang, K.-c. Lee, Z. Xu, Y. Qi, and Z. Ghahramani, "Distributed flexible nonlinear tensor factorization," *Advances in neural information processing systems*, vol. 29, 2016.

[9] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao, "Deep crossing: Web-scale modeling without manually crafted combinatorial features," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 255–262.

[10] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 17–37, 2017.

[11] F. Tanvir, M. I. K. Islam, and E. Akbas, "Predicting drug-drug interactions using meta-path based similarities," *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–8, 2021.

[12] F. Tanvir, K. M. Saifuddin, M. I. K. Islam, and E. Akbas, "Predicting drug-drug interactions using heterogeneous graph attention networks," *Proceedings of the 14th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263621744

[13] F. Tanvir, K. M. Saifuddin, M I. K. Islam, and E. Akbas, "Ddi prediction with heterogeneous information network - meta-path based approach," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1–12, 2024.

[14] F. Wan, L. Hong, A. Xiao, T. Jiang, and J. Zeng, "Neodti: Neural integration of neighbor information from a heterogeneous network for discovering new drug-target interactions," *bioRxiv*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49658098

[15] Y. Luo, X. Zhao, J. Zhou, J. Yang, Y. Zhang, W. Kuang, J. Peng, L. Chen, and J. Zeng, "A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information," *Nature Communications*, vol. 8, 2017.

[16] X. Chen, M.-X. Liu, and G.-Y. Yan, "Drug–target interaction prediction by random walk on the heterogeneous network," *Molecular BioSystems*, vol. 8, no. 7, pp. 1970–1978, 2012.

[17] G. Fu, Y. Ding, A. Seal, B. Chen, Y. Sun, and E. Bolton, "Predicting drug target interactions using meta-path-based semantic network analysis," *BMC Bioinformatics*, vol. 17, 2016.

[18] X. Zheng, H. Ding, H. Mamitsuka, and S. Zhu, "Collaborative matrix factorization with multiple similarities for predicting drug-target interactions," *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.

[19] H. Zhou, J. Li, J. Peng, S. Zhang, and S. Zhang, "Triplet attention: Rethinking the similarity in transformers," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2378–2388.

[20] G. Ji, K. Liu, S. He, and J. Zhao, "Distant supervision for relation extraction with sentence-level attention and entity descriptions," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[21] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[22] M. Žitnik and B. Zupan, "Data fusion by matrix factorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 1, pp. 41–53, 2014.

[23] J. Qu, B. Wang, Z. Li, X. Lyu, and Z. Tang, "Understanding multivariate drug-target-diseaseinterdependence via event-graph," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2021, pp. 1685–1687.

[24] K. M. Saifuddin, C. May, F. Tanvir, M. I. K. Islam, and E. Akbas, "Seq-hygan: Sequence classification via hypergraph attention network," *ArXiv*, vol. abs/2303.02393, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257365316

[25] K. M. Saifuddin, B. Bumgardnerr, F. Tanvir, and E. Akbas, "Hygnn: Drug-drug interaction prediction via hypergraph neural network," *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1503–1516, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250072419

[26] L. Liu, F. Huang, X. Liu, Z. Xiong, M. Li, C. Song, and W. Zhang, "Multi-view contrastive learning hypergraph neural network for drug-microbe-disease association prediction," in *International Joint Conference on Artificial Intelligence*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260845944

[27] System, "System, D. C. I. 2015. Smiles tutorial."

[28] K. Huang, C. Xiao, L. Glass, and J. Sun, "Explainable substructure partition fingerprint for protein, drug, and more," *NeurIPS Learning Meaningful Representation of Life Workshop*, 2019.

[29] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[30] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[31] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, pp. 455–500, 2009.

[32] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[33] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio, "Graph attention networks," *ArXiv*, vol. abs/1710.10903, 2018.

[34] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[35] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of the web conference 2020*, 2020, pp. 2704–2710.

[36] C. Moon, C. Jin, X. Dong, S. M. Abrar, W. Zheng, R. Y. Chirkova, and A. Tropsha, "Learning drug-disease-target embedding (ddte) from knowledge graphs to inform drug repurposing hypotheses," *Journal of biomedical informatics*, p. 103838, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235425831

[37] M. Li, X. Cai, S. Xu, and H. Ji, "Metapath-aggregated heterogeneous graph neural network for drug-target interaction prediction," *Briefings in bioinformatics*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:255462340

[38] K. Lawson, "A brief review of the pharmacology of amitriptyline and clinical outcomes in treating fibromyalgia," *Biomedicines*, vol. 5, no. 2, p. 24, 2017.

[39] P. Blier *et al.*, "Neurotransmitter targeting in the treatment of depression," *The Journal of clinical psychiatry*, vol. 74, no. suppl 2, p. 12763, 2013.

[40] C. M. Dempsey, S. M. Mackenzie, A. Gargus, G. Blanco, and J. Y. Sze, "Serotonin (5ht), fluoxetine, imipramine and dopamine target distinct 5ht receptor signaling to modulate caenorhabditis elegans egg-laying behavior," *Genetics*, vol. 169, no. 3, pp. 1425–1436, 2005.

[41] N. S. Philip, L. L. Carpenter, A. R. Tyrka, and L. H. Price, "Nicotinic acetylcholine receptors and depression: a review of the preclinical and clinical literature," *Psychopharmacology*, vol. 212, pp. 1–12, 2010.

HyGNN: Drug-Drug Interaction Prediction via Hypergraph Neural Network

Khaled Mohammed Saifuddin*, Briana Bumgardner¶, Farhan Tanvir§, Esra Akbas†
*† *Georgia State University*, USA
¶ *Rice University*, USA
§ *Oklahoma State University*, USA
*ksaifuddin1@student.gsu.edu, ¶bb64.edu@rice.edu, §farhan.tanvir@okstate.edu, †eakbas1@gsu.edu

2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00119

***Abstract*—Drug-Drug Interactions (DDIs) may hamper the functionalities of drugs, and in the worst scenario, they may lead to adverse drug reactions (ADRs). Predicting all DDIs is a challenging and critical problem. Most existing computational models integrate drug-centric information from different sources and leverage them as features in machine learning classifiers to predict DDIs. However, these models have a high chance of failure, especially for new drugs when all the information is not available. This paper proposes a novel *Hypergraph Neural Network* (`HyGNN`) model based on only the Simplified Molecular Input Line Entry System (SMILES) string of drugs, available for any drug, for the DDI prediction problem. To capture the drug chemical structure similarities, we create a hypergraph from drugs' chemical substructures extracted from the SMILES strings. Then, we develop `HyGNN` consisting of a novel attention-based *hypergraph edge encoder* to get the representation of drugs as hyperedges and a decoder to predict the interactions between drug pairs. Furthermore, we conduct extensive experiments to evaluate our model and compare it with several state-of-the-art methods. Experimental results demonstrate that our proposed `HyGNN` model effectively predicts DDIs and impressively outperforms the baselines with a maximum F1 score, ROC-AUC, and PR-AUC of 94.61%, 98.69%, and 98.68%, respectively. Finally, we show that our models also work well for new drugs.**

***Index Terms*—Drug-Drug Interaction, Graph Neural Network, Hypergraph, Hypergraph Neural Network, Hypergraph Edge Encoder**

I. Introduction

Many patients, especially those who suffer from chronic diseases such as high blood pressure, cancer, and heart failure, often consume multiple drugs concurrently for their disease treatment. Simultaneous usage of multiple drugs may result in Drug-Drug Interactions (DDIs). These interactions may unexpectedly reduce the efficacy of drugs and even may lead to adverse drug reactions (ADRs) [1], [2]. Therefore, it is important to identify potential DDIs early to minimize these adverse effects. However, since clinical trials to identify DDIs are performed on a few patients for a brief period [3], many potential new drug DDIs remain undiscovered before it is open to the market. Also, it is too expensive to do clinical experiments with all possible drug pairs. Thus, there is an obvious need for a computational model to detect DDIs and mitigate unanticipated reactions automatically.

With the availability of public databases, including drug-related information like DrugBank[1], STITCH[2], SIDER[3], PubChem[4], KEGG[5], etc., different computational models have been proposed to detect DDIs [4], [5]. Some of these models consider drug pairs' chemical structure SMILES similarity or binding properties [6]. SMILES is a specification that uses ASCII characters to define molecular structures explicitly. With a string of characters, SMILES may depict a three-dimensional chemical structure. On the other hand, out of the entire chemical structure, only a few substructures are responsible for chemical reactions between drugs, and the rest are less relevant [7]. However, considering the whole chemical structure may create a bias toward irreverent substructures and thus undermine the DDIs prediction [8]. With the increasing ability of more relational information about drugs, most of the current methods integrate multiple data sources to extract drug features such as side effects, target protein, pathways, and indications [9], [10].

Network-based methods have recently been explored in this domain, where drug networks are constructed based on drugs' known DDIs. Most of the graph-based methods consider a dyadic relationship between drugs. They operate on a simple regular graph where each vertex is a drug, and each edge shows a connection between two nodes. However, some methods also consider the relations of drugs to other biological entities to create heterogeneous graphs. Then, different topological information is extracted from the network to predict unknown links (i.e., interactions) between drugs. With the current advancement of the graph neural network (GNN), different GNN models for DDI prediction problems are proposed [5], [11]. While some of them create heterogeneous graphs manually from different resources, some of them create biomedical knowledge graphs by extracting triples from raw data (e.g., DrugBank) [12]–[14]. In these graphs, different entities, such as drugs, proteins, and side effects, are represented as nodes, and relations between those entities are represented as the edges. While including multiple drug-centric information from different sources would help to learn about DDI and have achieved strong performance, it is challenging to integrate

[1]https://go.drugbank.com/
[2]http://stitch.embl.de/
[3]http://sideeffects.embl.de/
[4]https://pubchem.ncbi.nlm.nih.gov/
[5]https://www.kegg.jp/

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00119

data from different resources. It is also tough to interpret which information is the most valuable in DDI prediction which needs a strong knowledge of biomedical entities and this is challenging for drugs in the early development stage. Moreover, multiple information may not always be available for all drugs, specially new drugs; therefore, these models may fail whenever any information is unavailable [15].

In this paper, to address these problems, we present a novel GNN-based approach for DDI prediction by only considering the SMILES string of the drugs, which is available for all drugs. Our method relies on the hypothesis that similar drugs behave similarly, are likely to interact with the same drugs, and two drugs are similar if they have similar substructures as functional groups in their SMILES strings [16], [17]. Finding similarities between SMILES strings based on their common substructures is a challenging task. To properly depict the structural-based similarity between drugs, we present them in a hypergraph setting, representing drugs as hyperedges connecting many substructures as nodes. A hypergraph is a unique model of a graph with hyperedges. Unlike a regular graph where the degree of each edge is 2, hyperedge is degree-free; it can connect an arbitrary number of nodes [18]–[21]. After constructing the hypergraph, we develop a Hypergraph Neural Network (`HyGNN`), a model that learns the DDIs by generating and using the representation of hyperedges as drugs. `HyGNN` has an encoder-decoder architecture. First, we present a novel *hypergraph edge encoder* to generate the embedding of drugs. Afterward, the pair-wise representations of drugs are passed through decoder functions to predict a binary score for each drug pair that represents whether two drugs interact. The main contributions of this paper are summarized as follows:

- **Hypergraph construction from SMILES strings:** We construct a novel hypergraph to depict the drugs' similarities. In the hypergraph, while each substructure extracted from the drugs' SMILES strings is represented as a node, each drug, consisting of a set of unique substructures, is represented as a hyperedge. This hypergraph represents the higher-level connections of substructures and drugs, which helps us to define complex similarities between chemical structures and drugs. Also, this helps us to learn better representation for drugs with GNN models with a passing message not only between 2 nodes but between many nodes and also between nodes and hyperedges.
- **Hypergraph GNN:** To learn and predict DDIs, we propose a novel hypergraph GNN model, called `HyGNN`, consisting of a novel hyperedge encoder and a decoder. Encoder exploits two layers where the first layer generates the embedding of nodes by aggregating the embedding of hyperedges. Then, the second layer generates the embedding of hyperedges (i.e., drugs) by aggregating the embedding of nodes. Since not all but a few substructures are mainly significant in chemical reactions, we use attention mechanisms to learn the significance of substructures (nodes) for drugs (edge) and chemical reactions. Furthermore, a decoder is modeled to predict the DDIs by taking the pair-wise drug representations as input. Our method solely utilizes drugs' chemical structure data to predict DDIs without requiring any other information or strong knowledge of biomedical entities. Chemical structures are obtained from SMILES strings that any drug has. Therefore, our model is applicable to any drugs, including new drugs, without other information, such as side effects and DDIs.
- **Extensive experiments:** We conduct extensive experiments to compare our proposed model with the state-of-the-art models. The results with different accuracy measures show that our method significantly outperforms all the baseline models. Also, we show with case studies that our model can find not only new DDIs for existing drugs but also DDIs for new drugs.

The rest of the paper is structured as follows. First, we briefly review the related work on DDI and hypergraph GNN in Section II. Section III describes our proposed `HyGNN` model, including hypergraph construction from SMILES strings, encoder and decoder layers of the `HyGNN` model works for generating the drug embedding and predicting DDIs. Next, in Section IV, we describe the experimental results and discussion. Finally, a conclusion is given in Section V.

II. Related Work

Many works have been proposed for the DDI prediction problem over the years. It can be categorized into similarity-based, classification-based, and network-based methods. Previous works assume that similar drugs have similar interaction profiles and define different similarities between drugs. Traditionally, pharmacological, topological, or semantic similarity based on statistical learning is utilized to predict DDIs. Vilar et al. [22] identify the DDIs based on molecular similarities. They represent each drug by a molecular fingerprint, a bit vector reflecting the presence/absence of a molecular feature. [23] develops INDI that uses seven different drug-drug similarity measures learned from drug side-effect, fingerprints, therapeutic effects, etc. Another vital research is [24] that incorporates four different biological information (e.g., target, transporter, enzyme, and carrier) of drugs to measure the similarity of drug pairs.

Some models extract features of drugs from various biological entities and drug interaction information and apply different machine learning (ML) methods for DDI training [8], [25]–[27]. Davazdahemami and Delen [26] constructed a graph containing both drug-protein and drug-side effect interactions. They also employed a classification method on the feature set and produced many similarities and centrality metrics based on the network. These features were then fed into four machine models. Luo et al. [27] propose a DDI prediction server that provides real-time DDI predictions based only on molecular structure. The server docks a drug's chemical structure across 611 human proteins to create a 611-dimensional docking vector. The drug pair features are created by concatenating the docking vectors for drug pairings. Finally, utilizing these features, a logistic regression model for DDI prediction is

developed. Ibrahim et al. [25] first extract different similarity features and employ logistic regression to pick the best feature; later, the best feature is used in six different ML classifiers to predict DDIs. One primary problem in DDI prediction is the lack of negative samples. A solution to address this problem is proposed in [8], named DDI-PULearn. It first generates negative samples using one-class SVM and kNN. Then positive and negative samples are used to predict DDIs.

Last decade, network-based models got great attention for drug-related problems. Some researchers construct a drug network using known DDIs where drugs are nodes, and interacting drugs are connected by a link [28]. Moreover, heterogeneous information networks leveraging different biomedical entities, such as proteins, side effects, pathways, etc., are also used to address similar problems [9]. As a different model, [29] constructs a molecular graph for each drug from its SMILES representation. Moreover, existing network-based models often extract drug embedding and directly learn latent node embedding using various embedding methodologies. As a result, their capacity to obtain specific neighborhood information on any organization in KG is restricted.

Recently, GNN has shown promising performance in different fields that include drug discovery [30], drug abuse detection [31], and drug-drug interaction [29], [32], [33], etc. Decagon [5] created a knowledge graph based on protein-protein, drug-drug, and drug-protein interactions. They also created a relational graph convolutional neural network for predicting multi-relational links in multimodal networks. Furthermore, they used a novel graph auto-encoder technique to create an end-to-end trainable model for link prediction on a multimodal graph. CASTER [7] recently created a dictionary learning framework for predicting DDIs based on drug chemical structures. They predict drug-drug interactions using the drug's molecular structure in a text format of SMILES [34] strings representation and outperform numerous deep learning approaches such as DeepDDI [35] and molVAE [36]. CASTER uses a sequential pattern mining approach to identify the common substrings included in the SMILES strings supplied during the training phase, which are then translated to an embedding using an encoder module. These features are then transformed into linear coefficients fed to a decoder and a predictor to yield DDI predictions. [33] constructs a drug network where two drugs are connected if they share common chemical substructures. Then, they apply different GNN models on the network to get the representation of drugs, and drug pair representation is passed to the ML classifier to predict interaction.

More recently, hypergraph and hypergraph neural network models have been developed to capture higher-order relations between different objects [19], [37]–[39]. All these works learn the representation of nodes and use these for node classification problems. Our HyGNN model is the first attempt to use hypergraph structure for DDI problems and, in general, drug-related problems. Also, our model aims to learn the representation of hyperedges and edge pair classification, which is different from current hypergraph neural network models.

III. HyGNN MODEL FOR DDI

In this section, we first define our DDI prediction problem and then summarize the preliminaries, model, and settings (Section III-A). Then we explain our hypergraph construction step with substructures extraction from Drugs (Section III-B). After that, we introduce our proposed HyGNN model with attention-based encoder and decoder layers (Section III-C).

A. Problem Formulation

Our goal is to develop a computational model that takes a drug pair (D_x, D_y) as input and predicts whether there exists an interaction between this drug pair. Each drug is represented by the SMILES string. SMILES is a unique chemical representation of a drug that consists of a sequence of symbols of molecules and the bonds between them.

Most of the graph-based existing DDI methods consider a dyadic relationship between drugs. This simple graph type considers an edge that can connect a maximum of two objects. However, there could be a more complex network in real life where an arbitrary number of nodes may interact as a group, so they could be connected through a hyperedge (i.e., triadic, tetradic, etc.). A hypergraph can be used to formulate such a complex network. A formal definition of the hypergraph is given below.

Hypergraph: *A hypergraph is a special kind of graph defined as $G = (V, E)$ where $V = \{v_1, ..., v_m\}$ is the set of nodes and $E = \{e_1.....e_n\}$ is the set of hyperedges. Each hyperedge e_j is degree-free and consists of an arbitrary number of nodes.*

Like the adjacency matrix of a regular graph, a hypergraph can be denoted by an incidence matrix H with $H_{i,j} = 1$ if the node i is in the hyperedge j as $v_i \in e_j$ and $H_{i,j} = 0$ otherwise.

In this paper, we construct a hypergraph network of drugs where each drug is a hyperedge, and the chemical substructures of drugs are the nodes. The chemical structures of a drug can be obtained from the SMILES, a unique chemical representation of a drug. We design a novel hypergraph neural networks model as an encoder-decoder architecture to accomplish the DDI prediction task. The encoder part exploits an attention mechanism to learn the representations of hyperedges (drugs) by giving attention to edges and nodes (substructures). The decoder predicts the interaction between drug pairs using latent learned drug features. The whole system is trained in a semi-supervised fashion. The functional architecture of this paper is shown in Figure 1. Our proposed model consists of two steps:

1) Hypergraph construction from SMILES,
2) DDI prediction with hypergraph neural networks
   a) Encoder: Drug (Hyperedge) representation learning
   b) Decoder: DDI prediction

B. Drug Hypergraph Construction

We construct a hypergraph to depict the structural similarities among drugs. At first, we decompose all the drugs'

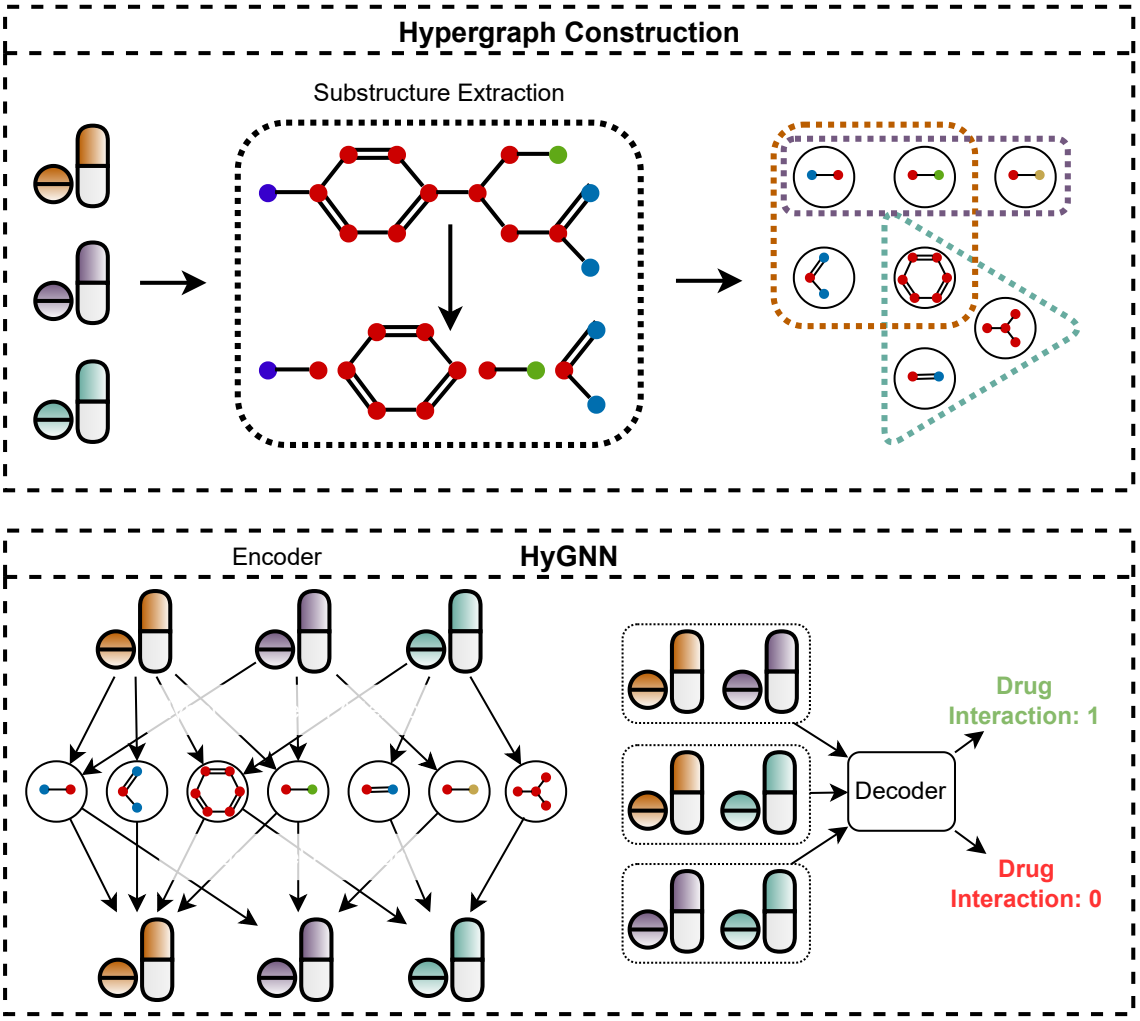


Fig. 1. System architecture of the proposed method. The First step is to construct a hypergraph network of drugs where each drug is a hyperedge, and frequent chemical substructures of drugs are the nodes. The second step is to design a hypergraph neural network (HyGNN) model with an attention-based encoder for hyperedge (drug) representation learning and decoder for DDI learning

SMILES into a set of substructures. In our drug hypergraph, these substructures are used as nodes. Moreover, each drug with a certain number of substructures is represented by a hyperedge. Drugs as hyperedges may connect with other drugs employing shared substructures as nodes. This hypergraph represents the higher-level connections of substructures and drugs, which may help to define complex similarities between chemical structures and drugs. Also, this helps us to learn better representation for drugs with GNN models with the passing message not only between 2 nodes but between many nodes and also between nodes and edges.

Substructures can be generated by utilizing different algorithms such as ESPF [40], k-mer [41], strobemers [42], etc. In this project, we use ESPF and k-mer to see the effect of substructures on the results. While k-mer use all extracted substructures, ESPF selects the most frequent ones. Algorithm 1 briefly shows the hypergraph construction steps.

ESPF: Like the concept of sub-word units in the natural language processing domain, ESPF is a powerful tool that

Algorithm 1: Drug Hypergraph Construction

Input: $SMILES\_strings$
Output: Hypergraph incident matrix: H
Call $Substructure\_Decom(SMILES\_strings)$;
`/* Substructure_Decom() could be ESPF or k-mer that decomposes SMILES into substructures. It returns a list of unique substructures and drug dictionary that will be used in the following for loop */`
for *each Substructure in Substructure_list* **do**
  if *Substructure is in Drug_dict[SMILES]* **then**
    $H[i,j] = 1$; `/* i,j is the id of substructure and drug, respectively. */`
  end
end
Output: H, Hypergraph incident matrix

Algorithm 2: Explainable Substructure Partition Fingerprint (ESPF)

Input: Set of initial SMILES tokens S as atoms and bonds, set of tokenized SMILES strings TS, frequency threshold α, and size threshold L for S.
for $t = 1 \ldots, L$ **do**
  $(S1, S2)$, $f \leftarrow$ scan TS ; /* $(S1, S2)$ is the frequentest consecutive tokens. */
  if $f < \alpha$ **then**
    break ; /* $(S1, S2)$'s frequency lower than threshold */
  end
  $TS \leftarrow$ find $(S1, S2) \in TS$, replace with $(S1S2)$; /* update TS with the combined token $(S1S2)$ */
  $S \leftarrow S \cup (S1S2)$; /* add $(S1S2)$ to the token vocabulary set S */
end
Output: TS, the updated tokenized drugs; S, the updated token vocabulary set.

decomposes sequential structures into interpretable functional groups. They consider that a few substructures are mainly responsible for drug chemical reactions, so they extract frequent substructures as influential ones. The ESPF algorithm is shown in Algorithm 2. Given a set of drug SMILES strings S, ESPF finds the frequent repetitive moderate-sized substructures from the set and replaces the original sequence with the substructures. If the frequency of each substructure in S is above a predefined threshold, it is added to a substructure list as a vocabulary. Substructures appear in this list in the most to least frequent order. We use this vocabulary list as our nodes and decompose any drug into a sequence of frequent substructures concerning those. For any given drug, we partition its SMILES in order of frequency, starting from the highest frequency. An example of partitioning a SMILES of a drug, DB00226, is as follows.

```
NC(N)=NCC1COC2(CCCCC2)O1
          ⇓
N C(N) =N CC1 CO C2 (CCC CC2) O1.
```

k-mer: k-mer is a tool to decompose sequential structures into subsequences of length k. It is widely used in biological sequence analysis and computational genomics. Similar to n-gram in natural language processing, a k-mer is a sequence of k characters in a string (or nucleotides in a DNA sequence). To get all k-mers from a sequence, we need to get the first k characters, then move just a single character to start the next k-mer, and so on. Effectively, this will create sequences that overlap in k-1 positions. Pseudocode for k-mer is shown in Algorithm 3.

For a sequence of length l, there are $l - k + 1$ numbers of k-mers and n^k total possible number of k-mers, where n is the number of monomers. k-mers are like words of a sentence.

Algorithm 3: k-mer

Input: $SMILES\_strings$, size threshold k
Substructure_list: []
Drug_dict:{}
for *each* $SMILES$ *in* $SMILES\_strings$ **do**
  Lst=[]
  for x *in range (l-k+1)* **do**
    /* l is the length of $SMILES$ */
    $C = SMILES[x : x + k]$
    $Lst.append(C)$
    $Substructure\_list.append(C)$
  end
  $Drug\_dict[SMILES] = Lst$
end
Output: Drug_dict, Substructure_list

k-mers help to bring out semantic features from a sequence. For example, for a sequence NCCO, monomers: {N, C, and O}, 2-mers: {NC, CC, CO} , 3-mers: {NCC, CCO}.

C. Hypergraph Neural Network (HyGNN) for DDI Prediction

We utilize the Hypergraph Neural Network (HyGNN) for DDI prediction. HyGNN includes an encoder, which generates the embedding of drugs, and a decoder that uses the embedding of drugs from the encoder to predict whether a drug pair interact or not.

*1) **Drug Representation learning via HyGNN - Encoder**:* To detect interacting pairs of drugs, we need features of drug pairs and, thus, features of drugs that encode their structure information. To generate features of drugs, we propose a novel *hypergraph edge encoder*. It creates $d^{'}$ dimensional embedding vectors for hyperedges (drugs) instead of nodes as in regular GNN models. Given the edge feature matrix, $F \in R^{E*d}$ and incidence matrix $H \in R^{V*E}$, the encoder of the HyGNN generates a feature vector of d' dimension through learning a function Z. Any layer (e.g., $(l+1)^{th}$ layer) of HyGNN can be expressed as

$$F^{l+1} = Z(F^l, H^T). \tag{1}$$

We consider the *hypergraph edge encoder* with a memory-efficient self-attention mechanism. It consists of two different levels of attention: (1) hyperedge-level attention, (2) node-level attention.

While hyperedge-level attention aggregates the hyperedge information to get the representation of nodes, node-level attention layer aggregates the connected vertex information to get the representation of hyperedges. In general, we define the HyGNN attention layers as

$$p_i^l = \mathbf{A_E}^l(p_i^{l-1}, q_j^{l-1} | \forall e_j \in E_i), \tag{2}$$

$$q_j^l = \mathbf{A_V}^l(q_j^{l-1}, p_i^l | \forall v_i \in e_j) \tag{3}$$

where $\mathbf{A_E}$ is an edge aggregator that aggregates features of hyperedges to get the representation p_i^l of node v_i in layer-l and E_i is the set of hyperedges that are connected to node v_i.

Similarly, $\mathbf{A_V}$ is a node aggregator that aggregates features of nodes to get the representation q_j^l of hyperedge e_j in layer-l and v_i is the node that connects to hyperedge e_j.

Hyperedge-level attention: In a hypergraph, each node may belong to multiple numbers of edges. However, the contribution of hyperedges to a node may not be equal. That is why we design an attention mechanism to highlight the crucial hyperedges and aggregate their features to compute the node feature p_i^l of node v_i. With the attention mechanism, p_i^l is defined as

$$p_i^l = \alpha\left(\sum_{e_j \in E_i} Y_{ij} W_1 q_j^{l-1}\right) \tag{4}$$

where α is a nonlinear activation function, W_1 is a trainable weight matrix that linearly transforms the input hyperedge feature into a high-level, E_i is the set of hyperedges connected to node v_i, and Y_{ij} is the attention coefficient of hyperedge e_j on node v_i. The attention coefficient is defined as

$$Y_{ij} = \frac{\exp(\mathbf{e_j})}{\sum_{e_k \in E_i} \exp(\mathbf{e_k})} \tag{5}$$

$$\mathbf{e_j} = \beta(W_2 q_j^{l-1} * W_3 p_i^{l-1}) \tag{6}$$

where β is a LeakyReLU activation function, W_2, W_3 are the trainable weight matrices, and $*$ is the element-wise multiplication.

Node-level attention: Each hyperedge in a hypergraph consists of an arbitrary number of nodes. However, the importance of nodes in a hyperedge construction may not be the same. We design a node-level attention mechanism to highlight a hyperedge's important nodes and aggregate their features accordingly to compute the hyperedge feature q_j^l of hyperedge e_j. With the attention mechanism, q_j^l is defined as

$$q_j^l = \alpha\left(\sum_{v_i \in e_j} X_{ji} W_4 p_i^l\right) \tag{7}$$

where W_4 is a trainable weight matrix, and X_{ji} is the attention coefficient of node v_i in the hyperedge e_j. The attention coefficient is defined as

$$X_{ji} = \frac{\exp(\mathbf{v_i})}{\sum_{v_k \in e_j} \exp(\mathbf{v_k})} \tag{8}$$

$$\mathbf{v_i} = \beta(W_5 p_i^l * W_6 q_j^{l-1}) \tag{9}$$

where v_k is the node that belongs to hyperedge e_j, W_5, W_6 are the trainable weight matrices.

Our hypergraph edge encoder model works based on these two attention layers that can capture high-order relations among data. Given the input hyperedge features, we first gather them to get the representation of nodes with hyperedge-level attention, then we gather the obtained node features to get the representation of hyperedges with node-level attention.

*2) **DDI prediction - Decoder**:* After getting the representation of drugs from the encoder layer, our target is to predict whether a given drug pair interacts or not. To accomplish this target, we design a decoder.

Given the vector representations (q_x, q_y) of drug pairs (D_x, D_y) as input, the decoder assigns a score, $p_{x,y}$ to each pair through a decoder function defined as

$$p_{x,y} = \gamma(q_x, q_y) \tag{10}$$

We use two different types of decoder functions:

MLP: After concatenating the features of drug pairs, we pass it through a multi-layer perceptron (MLP), which returns a scalar score for each pair

$$\gamma(q_x, q_y) = f_2(f_1(q_x \parallel q_y)) \tag{11}$$

where f_1 and f_2 are two different layers of MLP, and $||$ is the concatenation operation.

Dot product: We compute a scalar score for each edge by performing element-wise dot product between features of drug pairs using

$$\gamma(q_x, q_y) = q_x \cdot q_y \tag{12}$$

Afterward, we pass the decoder output through a sigmoid function $\sigma(\gamma(q_x, q_y))$ that generates predicted labels, Y', within the range 0 to 1. Any output value closer to 1 implies a high chance of interaction between two drugs.

*3) **Training the whole model**:* We consider the DDI prediction problem as a binary classification problem predicting whether there is an interaction between drug pairs or not. As a binary classification problem, we train our entire encoder-decoder architecture using a binary cross-entropy loss function defined as

$$loss = -\sum_{i=1}^{N} \left(Y_i \log Y_i^{'} + (1 - Y_i) \log(1 - Y_i^{'})\right) \tag{13}$$

where N is the total number of samples, Y_i is the actual label, and $Y_i^{'}$ is the predicted label.

D. Complexity

Our method is highly efficient with paralyzing across the edges and nodes [43]. The hyperedge encoder generates $d^{'}$ dimensional embedding vector for each hyperedge with a given initial feature as d dimensional vector using two-level attention. Hence, the time complexity in the encoder part is the cumulative complexity of attention layers. According to equation 4, the complexity in the hyperedge-level attention can be expressed as: $O(|E|dd^{'} + |V|Dd^{'})$, where D is the average degree of nodes. Similarly, the complexity in the node-level attention can be expressed as: $O(|V|dd^{'} + |E|Bd^{'})$, where B is the average degree of hyperedges.

IV. Experiment

In this section, we evaluate our proposed `HyGNN` model for DDI prediction with extensive experiments on two different datasets. We use F1, ROC-AUC, and PR-AUC accuracy

TABLE I
STATISTICS OF DATASET

Dataset	# of Drug	# of DDI
TWOSIDES	645	63473
DrugBank	1706	191402

metrics to compare our model's performances with the state-of-art baseline models. Making DDI predictions for new drugs could be more challenging than existing drugs. Therefore, we assess our model's performance for both new and existing drugs as well. First, we describe our datasets, TWOSIDES and DrugBank, then we explain our experiments, and present and analyze our results.

A. Dataset

We evaluate the proposed model using two different sizes of datasets. One is a small dataset, and another one is a large dataset. (1) **TWOSIDES** is our small dataset. **TWOSIDES** was created using data from adverse event reporting systems. Common adverse effects, such as hypotension and nausea, occur in more than a third of medication combinations, but others, such as amnesia and muscular spasms, occur in only a few. We extract 645 approved drugs' information from **TWOSIDES**. Each drug is linked to its chemical structure (SMILES). There are 63,473 DDI positive labels for the selected drugs. (2) **DrugBank** is our large dataset and it is the largest dataset for drugs that is publicly available. It is a drug knowledge database that includes clinical information about drugs, such as side effects and (DDIs). **DrugBank** also includes molecular data, such as the drug's chemical structure, target protein, and so on. From **DrugBank**, we retrieve information on 1706 approved drugs along with their SMILES strings and 191,402 DDI information. Both datasets are publicly available on Therapeutics Data Commons (TDC) [6]. The first unified platform, TDC, was launched to comprehensively access and assess machine learning across the entire therapeutic spectrum.

All known DDIs in both datasets are our positive samples. However, to train our model, we need negative samples as well. Therefore, we randomly sample a drug pair from the complement set of positive samples for each positive sample. Thus, we ensure a balanced dataset of equally positive and negative samples for an individual dataset.

We apply the ESPF algorithm and k-mer separately to extract the substructures from the SMILES string of drugs. For ESPF, we notice that when a lower frequency threshold is set, it generates many substructures, some of which may be unimportant. However, when a more significant threshold value is set, it generates fewer substructures and may lose some critical substructures. These substructures are used as nodes in the hypergraph. To examine the impact of the frequency threshold and thus the number of nodes in the hypergraph learning, we choose five different threshold values from 5 to 25. For k-mer, we notice that typically with the increment of k, the number of substructures (i.e., nodes) also increases. Similarly, to examine the impact of the k and thus the number of nodes in the hypergraph learning, we choose five different values of k from 3 to 15. A statistic of both datasets is shown in Table I. The number of nodes for different threshold values of ESPF and k-mer is given in Table II and Table III for each dataset.

[6]https://tdcommons.ai/

TABLE II
OF NODES (N) IN THE HYPERGRAPH BASED ON PARAMETERS OF THE METHODS, ESPF AND k-MER, FOR TWOSIDES DATASET

ESPF	$\|N\|$	k-mer	$\|N\|$
5	555	3	822
10	324	6	7025
15	249	9	14002
20	208	12	17351
25	187	15	18155

TABLE III
OF NODES (N) IN THE HYPERGRAPH BASED ON PARAMETERS OF THE METHODS, ESPF, AND k-MER, FOR DRUGBANK DATASET

ESPF	$\|N\|$	k-mer	$\|N\|$
5	1266	3	1296
10	729	6	11849
15	550	9	29443
20	462	12	43634
25	400	15	51315

B. Parameter Settings

Each dataset is randomly split into three parts: train (80%), validation (10%), and test (10%). We repeat this five times and report the average performances in terms of F1-score, ROC-AUC, and PR-AUC. The optimal hyper-parameters are obtained by grid search based on the validation set. The ranges of grid search are shown in Table IV.

We employ a single-layer `HyGNN` having two levels of attention. We use a LeakyReLU activation function in the encoder side and a ReLU activation function in the MLP predictor of the decoder side. During training, we simultaneously optimize the encoder and decoder using adam optimizer. Each model is trained for 2000 epochs with an early stop if there is no change in validation loss for 200 consecutive epochs.

For the baselines in subsection: IV-C, each GNN model is used as a two-layer architecture. All other parameters of each GNN are set by following their sources. For DeepWalk and node2vec, the walk length, number of walks, and window size are set to 100, 10, and 5, respectively. We use Logistic Regression as a simple ML classifier.

C. Baselines

We compare our model performance with different types of state-of-the-art models. We categorize the baseline models into five groups based on the data representation and methodology.

TABLE IV
HYPER-PARAMETER SETTINGS

Parameter	Values
Learning rate	1e-2, 5e-2, 1e-3, 5e-3
Hidden units	32, 64, 128
Dropout	0.1, 0.5
Weight decay	1e-2, 1e-3

1. Random walk-based embedding (RWE) on DDI graph We construct a regular graph based on the drug interaction information called a DDI graph. Drugs are represented as nodes, and two drugs share an edge if they interact. After constructing the graph, we apply the random walk-based graph embedding methods to get the representations of drugs. DeepWalk [44] and Node2vec [45] are two well-known graph embedding methods. They are both based on a similar mechanism of 'walk' on the graph traversing from one node to another. We apply DeepWalk and Node2Vec on the DDI graph and generate the embedding of nodes. Afterward, we concatenate drug embeddings to get the drug pair features and feed that into a machine learning classifier for binary classification.

2. GNN on DDI graph: After constructing the DDI graph as explained above, we apply three different GNN models with unsupervised settings; graph convolution network (GCN) [46], graph attention network (GAT) [43], and GraphSAGE [47] to get the representations of drugs. These GNN models are obtained from DGL[7]. After getting the representations of drugs, we concatenate them pair-wise and use them as the features of drug pairs in the ML classifier for binary classification.

3. GNN on substructure similarity graph (SSG) We follow [33] to create the substructure similarity graph (SSG). We construct an edge between two drugs if they have at least a predefined number of common substructures. We apply the ESPF algorithm to the SMILES strings of drugs to get the frequent substructures. Afterward, we apply three different GNN models, GCN, GAT, and GraphSAGE, to the constructed graph to get the representations of drugs. The drug representations are then concatenated pair-wise and fed into a classifier to predict the DDI.

4. CASTER We apply the Caster algorithm [7] for DDI prediction. It takes SMILES strings as input and employs frequent sequential pattern mining to discover the recurring substructures. They use the ESPF algorithm to extract frequent substructures. Then, they generate a functional representation for each drug using those frequent substructures. Further, the functional representation of drug pairs is used to predict DDIs. We reproduce CASTER results for our datasets.

[7]https://docs.dgl.ai/

5. Decagon Decagon [5] uses a multi-modal graph consisting of protein-protein interactions and drug-protein targets interactions for DDI prediction. It has an encoder-decoder architecture. The encoder exploits a graph convolution network to generate the representation of drugs by embedding all drug interactions with other entities in it. Then, the decoder takes drug pair representations as input and predicts DDIs with an exact side effect. The same TWOSIDES drug-drug interactions network is used in Decagon. That is why we directly compare our model performances with their reported results for TWOSIDES data instead of reproducing Decagon. However, we do not consider DrugBank data for Decagon as we do not have the additional information (e.g., side effects and target protein) in our DrugBank dataset to construct the multi-modal graph.

D. Results

1) Model Performances: We conduct detailed experiments on our proposed models for two different datasets with different threshold values of k-mer and ESPF. Both MLP and dot predictor-based decoder functions are employed individually for each setup to compare their performances. The overall performances are illustrated in Fig. 2 and Fig. 3. Fig. 2 depicts the model's performance for different ESPF frequency thresholds ranging from 5 to 25. This figure shows that it has a more significant impact on the TWOSIDES dataset, especially for the Dot decoder function than DrugBank. On TWOSIDES with MLP, it gives similar results till 25, and then it has a considerable decrease for 25. Since we get a significantly less number of substructures, it would not be enough to learn with those. For DrugBank, it gives similar results for different thresholds of ESPF. In general, frequency threshold 5 gives the best performance for TWOSIDES and DrugBank with MLP and DOT. As with the increment of the frequency threshold, the number of substructures (i.e., nodes) decreases, which could be a potential reason for performance degradation. The best performance for each dataset and decoder is recorded for a threshold value of 5.

Fig. 3 presents the models' performances for five different k values of k-mer ranges from 3 to 15. Similar to ESPF, the effect of the parameter on the results is higher for TWOSIDES than DrugBank. The reason for this could be that it is smaller than DrugBank, so the graph's size is affecting its results. However, DrugBank is a large dataset with enough training data to get good results, even with a small graph. Here, we can see that with the increment of the size of k-mer, the performance of the model increases, especially for TWOSIDES. As with increasing k, the number of substructures (i.e., nodes) increases which could be the reason for overall performance improvement. After some point, we will get too many substructures that could put noise into the data and decrease the model's performance. The best performance for each dataset and decoder are reported with $k = 9$ for k-mer.

2) Comparison with Baselines: We evaluate our models by comparing their performances with several baseline models and present the results in Table V for the TWOSIDES, and

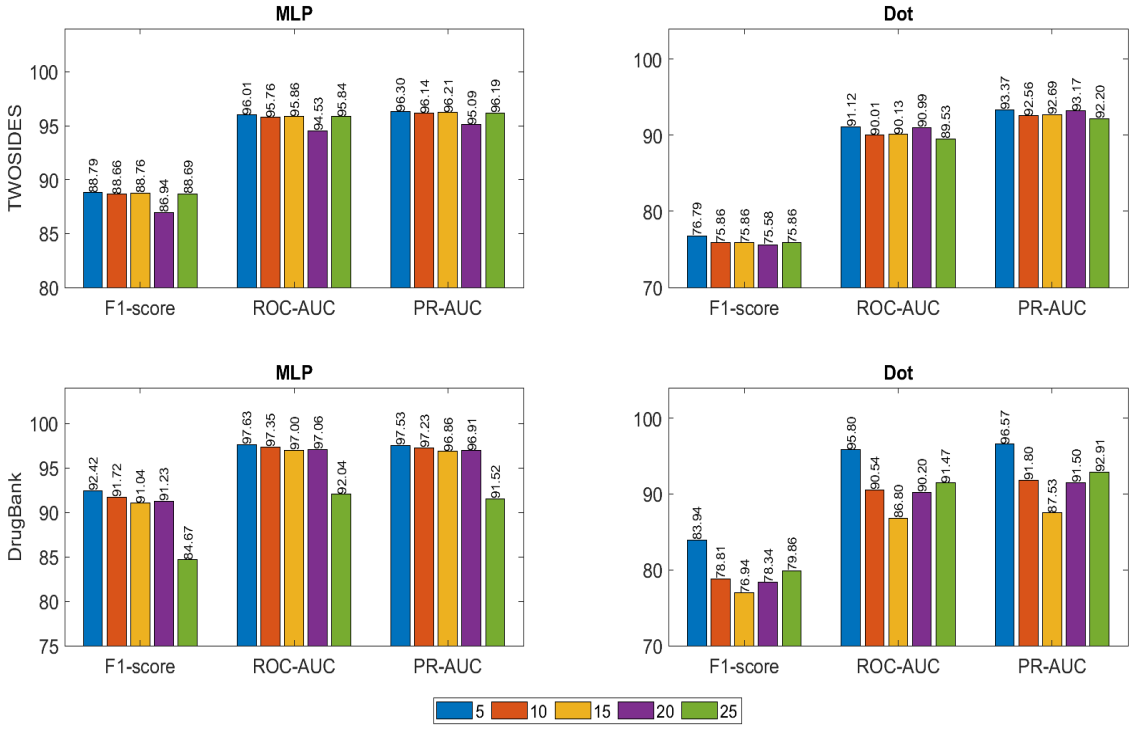


Fig. 2. Performance comparison of models for different frequency thresholds of ESPF.

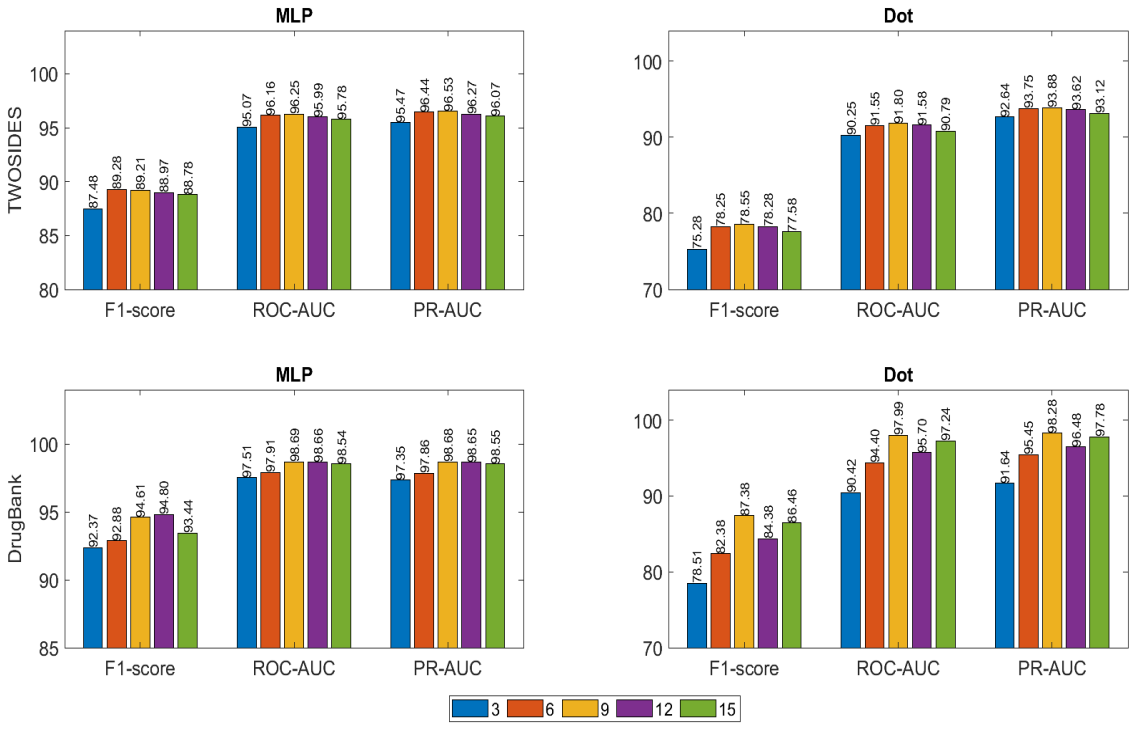


Fig. 3. Performance comparison of models for different sizes of k-mer.

TABLE V
PERFORMANCE COMPARISONS OF HyGNN WITH BASELINE MODELS ON TWOSIDES DATASET.

Model	Method	F1	ROC-AUC	PR-AUC
RWE on DDI Graph	DeepWalk	80.35	80.36	85.19
	Node2vec	84.50	84.52	88.33
GNN on DDI graph	GCN	85.34	85.38	88.87
	GraphSage	85.83	85.80	89.28
	GAT	82.67	82.68	86.86
GNN on SSG graph	GCN	53.85	54.04	66.94
	GraphSage	60.19	60.18	70.34
	GAT	54.25	54.37	66.85
CASTER	-	82.35	90.45	90.58
Decagon	-	-	87.20	83.20
HyGNN	ESPF & MLP	88.79	96.01	96.30
	ESPF & Dot	76.79	91.12	93.37
	k-mer & MLP	**89.21**	**96.25**	**96.53**
	k-mer & Dot	78.55	91.80	93.88

TABLE VI
PERFORMANCE COMPARISONS OF HyGNN WITH BASELINE MODELS ON DRUGBANK DATASET.

Model	Method	F1	ROC-AUC	PR-AUC
RWE on DDI Graph	DeepWalk	73.34	73.35	80.05
	Node2vec	79.52	79.54	84.56
GNN on DDI graph	GCN	77.05	77.06	82.78
	GraphSage	80.83	80.88	85.51
	GAT	63.84	69.75	78.52
GNN on SSG graph	GCN	58.00	58.04	69.11
	GraphSage	61.10	61.15	70.64
	GAT	58.20	58.24	69.25
CASTER	-	87.36	94.27	94.20
HyGNN	ESPF & MLP	92.42	97.63	97.53
	ESPF & Dot	83.94	95.80	96.57
	k-mer & MLP	**94.61**	**98.69**	**98.68**
	k-mer & Dot	87.38	97.99	98.28

Table VI for the DrugBank dataset. As we see in these tables, our models comprehensively outperform all the baseline models. More precisely, in Table V for the TWOSIDES dataset, HyGNN achieves at least 7% on F1, 6% on ROC-AUC, and PR-AUC better performance than other baseline models. While the best model among all the baselines, CASTER, achieves an F1 score of 82.35%, our HyGNN with k-mer & MLP scores 89.21% with almost 7% gain. A similar situation also happens for the other two accuracy measures: ROC-AUC and PR-AUC.

In Table V, for the DDI graph, from the GNN models, GraphSage gives the best results with 85.83%, 85.80%, and 89.29% on F1, ROC-AUC, and PR-AUC, respectively. Also, from random walk-based embedding models, Node2Vec gives the best results, which is very similar to GraphSage results with 84.50% on F1, 84.52% on ROC-AUC, and 88.33% PR-AUC scores. For the SSG graph, again, GraphSage gives the best result. In our result comparison table, CASTER is the best competitor of HyGNN. Out of all baseline models, CASTER shows the best performance with ROC-AUC and PR-AUC of above 90%. Decagon is a multi-modal graph that also exhibits better performance than GNN on SSG.

Table VI presents the performance comparison of HyGNN with baselines for the DrugBank dataset. As for TWOSIDES, here again, out of three different GNN models on DDI and SSG graphs, GraphSage yields the best result. Similarly, Node2Vec performs better than DeepWalk. CASTER is still the best performer among all baselines, with 87.36%, 94.27%, and 94.20% on F1, ROC-AUC, and PR-AUC, respectively. However, our HyGNN with k-mer & MLP significantly surpasses CASTER with 94.61% on F1, 98.69% on ROC-AUC, and 98.68% on PR-AUC. As Decagon depends on the drug and other drug-centric information, we could not experiment with it on the DrugBank dataset.

In summary, HyGNN with k-mer gives better results than ESPF. The reason for this could be that with the ESPF, we eliminate many substructures but keep just frequent ones. This

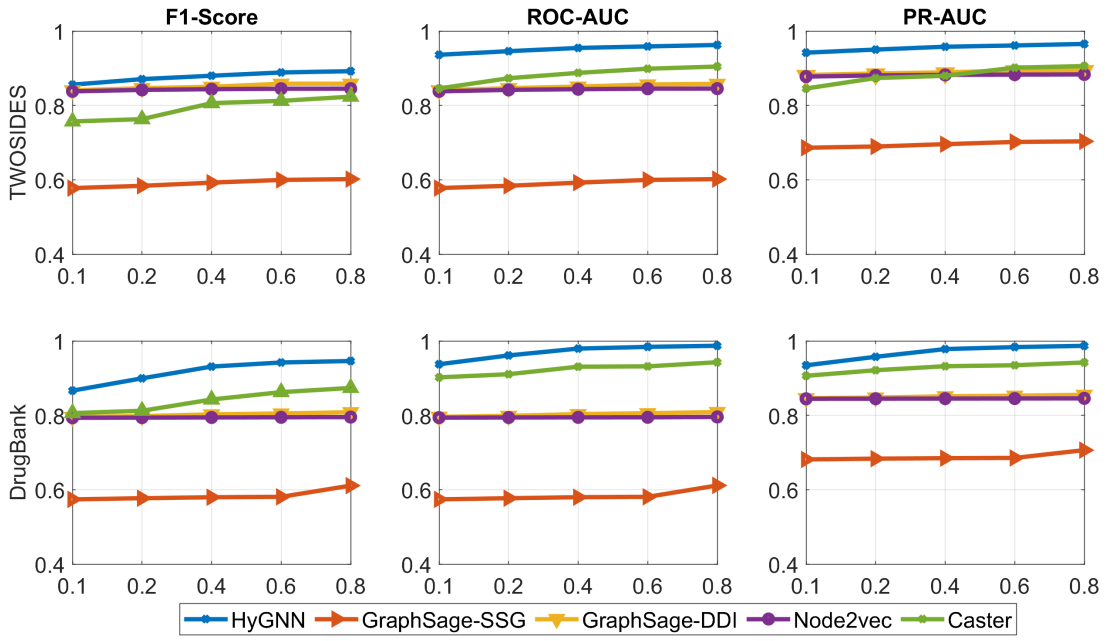


Fig. 4. Performance comparison of models for different training sizes where x-axes represent the training percentages.

may result in losing important ones that are not frequent. However, with k-mer, we get all and let the attention models in HyGNN learn which substructures are more important for DDI.

Moreover, we take the best-performing method from each baseline model, namely Node2Vec from random walk-based embedding, GraphSage from GNN on DDI, GraphSage from GNN on SSG, CASTER, and k-mer & MLP from our HyGNN models. Then, we compare their performances by changing the training sizes from 10% to 80% for both datasets. A comparison of performance is outlined in Fig 4. Results indicate HyGNN to be the best-performing model, and it still gives very good results with small training data. However, decreasing the training size affects the baseline models significantly, especially GraphSage on the SSG model. It is worthy of mention that based on our results, all graph-based models, especially different variants of GNNs, including HyGNN and baselines, have performed fairly well on our data.

Hypergraphs are used in a wide range of scientific fields. Hypergraphs are a natural method to illustrate shared group relationships. Through a hypergraph structure, HyGNN is able to capture higher-order correlations between data (i.e., triadic, tetradic, etc.). Furthermore, employing an attention mechanism makes it more robust by giving more weight to important substructures while learning representations of drugs. Though GAT has attention architecture as well, it can not discover the important edges. The main strength of our HyGNN is the proposed *hypergraph edge encoder* that has two levels of attention mechanism. At first, it aggregates the hyperedges to generate the representation of the node. While aggregating, it imposes more attention on the important hyperedges. Similarly, to generate the representation of a hyperedge, it aggregates the nodes' information with much attention to the important ones.

Moreover, HyGNN has a decoder function, and we learn all the parameters of the encoder and decoder simultaneously during training. From Table V and Table VI, we can see HyGNN with k-mer & MLP performs better than dot product. k-mers are k-length substrings included inside a biological sequence. A bigger k-mer is preferable since it ensures greater uniqueness in the base sequences that will create the string. Larger k-mer sizes aid in the elimination of repetitive substrings. Moreover, MLP Predictors are well-suited for classification problems in which data is labeled. They are extremely adaptable and may be used to learn a mapping from inputs to outputs in general. Additionally, it generates superior results compared to dot predictor since it has trainable parameters that are learned throughout the training.

3) Case Study - Prediction and Validation of Novel DDIs: We evaluate the effectiveness of HyGNN model on novel DDIs prediction. We select some drug pairs from TWOSIDES. None of these drug pairs have DDI info in TWOSIDES but have DDI info in DrugBank. Then we train our HyGNN using TWOSIDES and make predictions for those pairs. Following that, we collect predicted scores for those drug pairs as shown in Table VII. From this table, we see that for the first eight drug pairs, though the TWOSIDES label for each of these pairs is zero, we get predicted scores above 90% for each pair, which shows there is a high chance that each pair will interact between them. To further validate it, we cross-check our predicted score with DrugBank, which says each of these eight drug pairs interacts between them. Moreover, for the last four-drug pairs of Table VII the predicted scores are minimal,

TABLE VII
NOVEL DDI PREDICTIONS BY HYGNN ON TWOSIDES DATASET

Drug1	Drug2	TWOSIDES Label	Predicted DDI Score	DrugBank Label
Desvenlafaxine	Paroxetine	0	0.9989	1
Probenecid	Metformin	0	0.9931	1
Fluvastatin	Metronidazole	0	0.9212	1
Loratadine	Isradipine	0	0.9703	1
Glyburide	Bosentan	0	0.9068	1
Salmeterol	Dicycloverine	0	0.9189	1
Valdecoxib	Sodium sulfate	0	0.9105	1
Lisinopril	Naratriptan	0	0.9336	1
Bexarotene	Maprotiline	0	9.9993e-10	0
Amoxapine	Econazole	0	6.8256e-09	0
Nabilone	Oxaprozin	0	4.1440e-08	0
Dexmedetomidine	Carbachol	0	1.2417e-08	0

TABLE VIII
NOVEL DDI PREDICTIONS BY HYGNN ON DRUGBANK DATASET

Drug1	Drug2	DrugBank Label	Predicted DDI Score	TWOSIDES Label
Hydroxychloroquine	Loratadine	0	0.9879	1
Dextromethorphan	Ofloxacin	0	0.9772	1
Midazolam	Warfarin	0	0.9884	1
Benzthiazide	Fentanyl	0	5.6989e-14	0
Labetalol	Levonorgestrel	0	9.1049e-07	0
Cefprozil	Disulfiram	0	1.0882e-11	0

TABLE IX
PERFORMANCE OF HYGNN FOR NEW DRUGS

Dataset	Unseen Node	F1	ROC-AUC	PR-AUC
TWOSIDES	5%	72.75	78.25	85.64
DrugBank	5%	65.23	70.84	78.04

and TWOSIDES, and DrugBank both say they don't interact. Similarly, six drug pairs are selected from DrugBank having no DDI info in DrugBank but in TWOSIDES as shown in Table VIII, then HyGNN is trained using DrugBank data and validated the predicted scores by TWOSIDES.

4) Case Study- DDI Prediction for New Drugs: Making DDI predictions for new drugs could be more challenging than existing drugs. Since the model does not learn based on the SMILES strings of new drugs. To show the effectiveness of our model for new drugs, at first, we randomly select a 5% drug from a dataset and completely remove these drugs' information from the corresponding train set and keep those drugs' information only in the test set. These selected 5% drugs can be considered new drugs. The experimental results for both datasets with new drugs are shown in Table IX. As we see in the table, our model predicts DDIs effectively for both datasets.

V. CONCLUSION

In this paper, we propose a novel GNN-based framework for DDI prediction based on the chemical structures (SMILES) of drugs. In contrast to existing graph-based models, we utilize a novel hypergraph structure to depict higher-order structural similarities between drugs. Following that, we propose a Hypergraph GNN model with an encoder-decoder architecture to learn the drug representation for DDI prediction. We develop a hypergraph edge encoder to construct drug embeddings and a decoder with drug representations to predict a score for each drug pair, indicating whether two drugs interact. Finally, with several experiments, we show that our method outperforms different types of baseline models and also it is able to predict DDIs for new drugs.

As future work, we plan to extend our model to address other problems in bioinformatics, like protein-protein interaction prediction, drug repurposing, and sequence classification.

ACKNOWLEDGMENT

This work is funded partially by National Science Foundation under Grant No 2104720.

REFERENCES

[1] Ke Han, Peigang Cao, Yu Wang, Fang Xie, Jiaqi Ma, Mengyao Yu, Jianchun Wang, Yaoqun Xu, Yu Zhang, and Jie Wan. A Review of

Approaches for Predicting Drug–Drug Interactions Based on Machine Learning. *Frontiers in Pharmacology*, 12:3966, jan 2022.

[2] Yanchao Tan, Chengjun Kong, Leisheng Yu, Pan Li, Chaochao Chen, Xiaolin Zheng, Vicki S Hertzberg, and Carl Yang. 4sdrug: Symptom-based set-to-set small and safe drug recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3970–3980, 2022.

[3] Yang Qiu, Yang Zhang, Yifan Deng, Shichao Liu, and Wen Zhang. A Comprehensive Review of Computational Methods for Drug-drug Interaction Detection. *IEEE/ACM transactions on computational biology and bioinformatics*, PP, 2021.

[4] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. Graph representation learning in bioinformatics: trends, methods and applications. *Briefings in Bioinformatics*, 23(1):bbab340, 2022.

[5] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, jul 2018.

[6] Takako Takeda, Ming Hao, Tiejun Cheng, Stephen H. Bryant, and Yanli Wang. Predicting drug-drug interactions through drug structural similarities and interaction networks incorporating pharmacokinetics and pharmacodynamics knowledge. *Journal of Cheminformatics*, 9(1):1–9, mar 2017.

[7] Kexin Huang, Cao Xiao, Trong Hoang, Lucas Glass, and Jimeng Sun. Caster: Predicting drug interactions with chemical substructure representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 702–709, 2020.

[8] Yi Zheng, Hui Peng, Xiaocai Zhang, Zhixun Zhao, Xiaoying Gao, and Jinyan Li. DDI-PULearn: A positive-unlabeled learning method for large-scale prediction of drug-drug interactions. *BMC Bioinformatics*, 20(19):1–12, dec 2019.

[9] Farhan Tanvir, Muhammad Ifte Khairul Islam, and Esra Akbas. Predicting Drug-Drug Interactions Using Meta-path Based Similarities. In *2021 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. Institute of Electrical and Electronics Engineers (IEEE), oct 2021.

[10] Yang Qiu, Yang Zhang, Yifan Deng, Shichao Liu, and Wen Zhang. A comprehensive review of computational methods for drug-drug interaction detection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2021.

[11] Yue Hua Feng, Shao Wu Zhang, and Jian Yu Shi. DPDDI: a deep predictor for drug-drug interactions. *BMC bioinformatics*, 21(1):419, sep 2020.

[12] Zhong-Hao Ren, Chang-Qing Yu, Li-Ping Li, Zhu-Hong You, Yong-Jian Guan, Xin-Fei Wang, and Jie Pan. Biodkg–ddi: predicting drug–drug interactions based on drug knowledge graph fusing biochemical information. *Briefings in Functional Genomics*, 21(3):216–229, 2022.

[13] Yue Yu, Kexin Huang, Chao Zhang, Lucas M Glass, Jimeng Sun, and Cao Xiao. Sumgnn: multi-typed drug interaction prediction via efficient knowledge graph summarization. *Bioinformatics*, 37(18):2988–2995, 2021.

[14] Yue Yu, Kexin Huang, Chao Zhang, Lucas M Glass, Jimeng Sun, and Cao Xiao. SumGNN: Multi-typed Drug Interaction Prediction via Efficient Knowledge Graph Summarization. *Bioinformatics (Oxford, England)*, 37(18):2988–2995, sep 2021.

[15] Suyu Mei and Kun Zhang. A machine learning framework for predicting drug–drug interactions. *Scientific Reports 2021 11:1*, 11(1):1–12, sep 2021.

[16] Santiago Vilar, Rave Harpaz, Eugenio Uriarte, Lourdes Santana, Raul Rabadan, and Carol Friedman. Drug—drug interaction through molecular structure similarity analysis. *Journal of the American Medical Informatics Association*, 19(6):1066–1074, 2012.

[17] Yvonne C Martin, James L Kofron, and Linda M Traphagen. Do structurally similar molecules have similar biological activity? *Journal of medicinal chemistry*, 45(19):4350–4358, 2002.

[18] Alain Bretto. Hypergraph theory. *An introduction. Mathematical Engineering. Cham: Springer*, 2013.

[19] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph Neural Networks. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 3558–3565, sep 2018.

[20] Mehmet Emin Aktas and Esra Akbas. Hypergraph laplacians in diffusion framework. In *International Conference on Complex Networks and Their Applications*, pages 277–288. Springer, 2021.

[21] Mehmet Emin Aktas, Thu Nguyen, Sidra Jawaid, Rakin Riza, and Esra Akbas. Identifying critical higher-order interactions in complex networks. *Scientific reports*, 11(1):1–11, 2021.

[22] Santiago Vilar, Rave Harpaz, Eugenio Uriarte, Lourdes Santana, Raul Rabadan, and Carol Friedman. Drug-drug interaction through molecular structure similarity analysis. *Journal of the American Medical Informatics Association*, 19(6):1066–1074, nov 2012.

[23] Assaf Gottlieb, Gideon Y. Stein, Yoram Oron, Eytan Ruppin, and Roded Sharan. INDI: a computational framework for inferring drug interactions and their associated recommendations. *Molecular Systems Biology*, 8:592, 2012.

[24] Reza Ferdousi, Reza Safdari, and Yadollah Omidi. Computational prediction of drug-drug interactions based on drugs functional similarities. *Journal of biomedical informatics*, 70:54–64, jun 2017.

[25] Heba Ibrahim, Ahmed M. El Kerdawy, A. Abdo, and A. Sharaf Eldin. Similarity-based machine learning framework for predicting safety signals of adverse drug–drug interactions. *Informatics in Medicine Unlocked*, 26:100699, jan 2021.

[26] Behrooz Davazdahemami and Dursun Delen. A chronological pharmacovigilance network analytics approach for predicting adverse drug events. *Journal of the American Medical Informatics Association*, 25:1311–1321, 2018.

[27] Heng Luo, Ping Zhang, Hui Huang, Jialiang Huang, Emily Kao, Leming Shi, Lin He, and Lun Yang. Ddi-cpi, a server that predicts drug–drug interactions through implementing the chemical–protein interactome. *Nucleic Acids Research*, 42:W46 – W52, 2014.

[28] Andrej Kastrin, Polonca Ferk, and Brane Leskošek. Predicting potential drug-drug interactions on topological and semantic similarity features using statistical learning. *PLoS ONE*, 13(5), may 2018.

[29] Xin Chen, Xien Liu, and Ji Wu. Drug-drug Interaction Prediction with Graph Representation Learning. *Proceedings - 2019 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2019*, pages 354–361, nov 2019.

[30] Thomas Gaudelet, Ben Day, Arian R. Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B.R. Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L. Blundell, Michael M. Bronstein, and Jake P. Taylor-King. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 22(6):1–22, nov 2021.

[31] Khaled Mohammed Saifuddin, Muhammad Ifte Khairul Islam, and Esra Akbas. Drug Abuse Detection in Twitter-sphere: Graph-Based Approach. *2021 IEEE International Conference on Big Data (Big Data)*, pages 4136–4145, jan 2021.

[32] Yunsheng Bai, Ken Gu, Yizhou Sun, and Wei Wang. Bi-Level Graph Neural Networks for Drug-Drug Interaction Prediction. In *arXiv preprint arXiv:2006.14002*, jun 2020.

[33] Bri Bumgardner, Farhan Tanvir, Khaled Mohammed Saifuddin, and Esra Akbas. Drug-Drug Interaction Prediction: a Purely SMILES Based Approach. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5571–5579. Institute of Electrical and Electronics Engineers (IEEE), jan 2022.

[34] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28:31–36, 1988.

[35] Jae Yong Ryu, Hyun Uk Kim, and Sang Yup Lee. Deep learning improves prediction of drug–drug and drug–food interactions. *Proceedings of the National Academy of Sciences*, 115:E4304 – E4311, 2018.

[36] Rafael Gómez-Bombarelli, David Kristjanson Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4:268 – 276, 2018.

[37] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be More with Less: Hypergraph Attention Networks for Inductive Text Classification. *EMNLP 2020 - 2020 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 4927–4936, nov 2020.

[38] Chaofan Chen, Zelei Cheng, Zuotian Li, and Manyi Wang. Hypergraph attention networks. *Proceedings - 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2020*, pages 1560–1565, dec 2020.

[39] Hyunjin Hwang, Seungwoo Lee, and Kijung Shin. HyFER: A Framework for Making Hypergraph Learning Easy, Scalable and Benchmarkable. 2021.
[40] Kexin Huang, Cao Xiao, Lucas Glass, and Jimeng Sun. Explainable Substructure Partition Fingerprint for Protein, Drug, and More. In *NeurIPS Learning Meaningful Representation of Life Workshop*, 2019.
[41] Jingsong Zhang, Jianmei Guo, Xiaoqing Yu, Xiangtian Yu, Weifeng Guo, Tao Zeng, and Luonan Chen. Mining K-mers of Various Lengths in Biological Sequences. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10330 LNBI:186–195, 2017.
[42] Kristoffer Sahlin. Strobemers: an alternative to k-mers for sequence comparison. *bioRxiv*, page 2021.01.28.428549, apr 2021.
[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
[44] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
[45] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
[46] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
[47] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.