Seq-HyGAN: Sequence Classification via Hypergraph Attention Network

Khaled Mohammed Saifuddin*
Georgia State University
Atlanta, Georgia, USA
ksaifuddin1@student.gsu.edu

Corey May
Arkansas Tech University
Russellville, Arkansas, USA
cmay12@atu.edu

Farhan Tanvir
Oklahoma State University
Stillwater, Oklahoma, USA
farhan.tanvir@okstate.edu

Muhammad Ifte Khairul Islam
Georgia State University
Atlanta, Georgia, USA
mislam29@student.gsu.edu

Esra Akbas
Georgia State University
Atlanta, Georgia, USA
eakbas1@gsu.edu

ABSTRACT

Extracting meaningful features from sequences and devising effective similarity measures are vital for sequence data mining tasks, particularly sequence classification. While neural network models are commonly used to automatically learn sequence features, they are limited to capturing adjacent structural connection information and ignoring global, higher-order information between the sequences. To address these challenges, we propose a novel Hypergraph Attention Network model, namely Seq-HyGAN for sequence classification problems. To capture the complex structural similarity between sequence data, we create a novel hypergraph model by defining higher-order relations between subsequences extracted from sequences. Subsequently, we introduce a Sequence Hypergraph Attention Network that learns sequence features by considering the significance of subsequences and sequences to one another. Through extensive experiments, we demonstrate the effectiveness of our proposed Seq-HyGAN model in accurately classifying sequence data, outperforming several state-of-the-art methods by a significant margin.

CCS CONCEPTS

• **Information systems** → *Data mining*; *Computing platforms.*

KEYWORDS

Hypergraph Attention Network, Hypergraph Neural Network, Sequence Learning, Graph Learning, Attention Network

ACM Reference Format:
Khaled Mohammed Saifuddin, Corey May, Farhan Tanvir, Muhammad Ifte Khairul Islam, and Esra Akbas. 2023. Seq-HyGAN: Sequence Classification via Hypergraph Attention Network. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, UK*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3615057

CIKM '23, October 21–25, 2023, Birmingham, UK

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615057

1 INTRODUCTION

Extracting meaningful features from sequences and devising effective similarity measures are vital for sequence data mining tasks, particularly sequence classification. Neural networks (NN), especially recurrent neural networks (RNN) (i.e., LSTM, GRU), are commonly used to learn features capturing the adjacent structural information [28, 37, 38]. However, these models may struggle to capture non-adjacent, high-order, and complex relationships present in the sequence data. Recently, graph has also been explored for different types of sequence data classification tasks such as text classification [34], DNA-protein binding prediction [18], protein function prediction [15], drug-drug interaction prediction [8], etc. Graphs, as highly sophisticated data structures, have the capability to effectively capture both local and global non-adjacent information within the data [23, 30]. The state-of-the-art models for sequence-to-graph conversion can be classified into two main categories: order-based graphs and similarity-based graphs [5, 11, 48].

While existing graph models for sequence data achieved great performance, they still have some key challenges. Order-based models generate many large and sparse graphs, especially when dealing with a long and large number of sequences. Handling such many large graphs can lead to increased computational and memory requirements. Similarly, similarity-based graphs encounter challenges when calculating similarities between all pairs of sequences, especially for large datasets. Furthermore, order-based graphs fail to capture relationships between sequences beyond the intra-sequence connections with considering dyadic relationships between nodes. However, sequence data may possess more complex relationships, such as triadic or tetradic relationships, which these models are unable to capture. The selection of appropriate similarity measures becomes problematic, and using similarity values to define relationships between sequences can lead to information loss.

In order to overcome the aforementioned challenges for sequence classification, we propose a novel Hypergraph Attention Network model, namely Seq-HyGAN. Our approach is built on the assumption that sequences sharing structural similarities tend to belong to the same classes, and sequences can be considered similar if they contain similar subsequences. To effectively capture the structural similarities between sequences, we represent them in a hypergraph framework, where the sequences are depicted as hyperedges that connect their respective subsequences as nodes. This construction

allows us to create a single hypergraph encompassing all the sequences in the dataset. Unlike a standard graph where the degree of each edge is 2, hyperedge is degree-free; it can connect an arbitrary number of nodes [1, 2, 7]. To enhance the representation of sequences and capture complex relationships among them, we introduce a novel Seq-HyGAN architecture that employs a three-level attention-based neural network. Unlike regular NNs (i.e., RNN) that can only capture local information, Seq-HyGAN is more robust as it can capture both local (within the sequence) and global (between sequences) information from the sequence data. Moreover, while traditional Graph Neural Networks (GNNs) with standard graphs are limited to a message-passing mechanism between two nodes, this hypergraph setting assists GNNs in learning a much more robust representation of sequences with a message-passing mechanism not only between two nodes but between many nodes and also between nodes and hyperedges.

Our contributions are summarized as follows:

- **Hypergraph construction from sequences:** We introduce a novel hypergraph construction model from the sequence dataset. In our proposed sequence hypergraph, each subsequence extracted from the sequences is represented as a node, while each sequence, composed of a unique set of subsequences, is represented as a hyperedge. By leveraging this hypergraph model, we can capture and define higher-order complex structural similarities that exist between sequences.
- **Hypergraph Attention Network:** We introduce a novel hypergraph attention network model, Seq-HyGAN, specifically designed for sequence classification tasks. Our model focuses on learning sequence representations as hyperedges while considering both local and global context information with three levels of aggregation with attention that captures different levels of context. At the first level, it generates node embedding that incorporates global context by aggregating hyperedge embeddings. At the second level, the model refines node embeddings for each hyperedge. It captures local context by aggregating neighboring node embeddings in the same hyperedge and considering the subsequence position in a sequence. Finally, at the third level, it generates sequence embedding by aggregating node embeddings from both global (level 1) and local (level 2) perspectives, resulting in a comprehensive representation of the sequences.
- **Capturing importance via attention:** To capture the relative importance of individual subsequences (nodes) within each sequence (hyperedge), our model incorporates an attention mechanism. This mechanism allows the model to learn the varying significance of specific subsequences in contributing to the overall similarity between sequences. Additionally, the attention mechanism enables the model to discern the importance of both subsequences and sequences in relation to each subsequence. By doing so, our model comprehensively captures the inter-dependencies between different levels of granularity in the data, facilitating a nuanced understanding of the structure of the data.
- **Extensive experiments:** We conduct extensive experiments to show the effectiveness of our model on four different datasets and five different classification problems. We also compare the proposed Seq-HyGAN model with the state-of-the-art baseline models. The results with different accuracy measures show that our method significantly surpasses the baseline models.

The rest of this paper is organized as follows. In Section 2, we provide a concise overview of the existing literature on sequence classification and hypergraph GNNs. Section 3 presents the proposed Seq-HyGAN model, outlining the process of constructing a hypergraph from a sequence dataset and the details of the proposed hypergraph attention network. Subsequently, Section 4 presents the results obtained from our extensive experiments. Finally, Section 5 concludes the paper by summarizing the key findings.

2 RELATED WORK

Various studies have been carried out on the problem of sequence classification. We broadly categorize them into three different types of methods: Machine Learning (ML)-based, Deep Learning (DL)-based, and GNN-based. ML-based methods generate a feature vector using some kernel function such as k-spectrum kernel [26], local alignment kernel [35], and then they apply ML classifiers for sequence classification tasks. Applications of DL-based methods, especially recurrent neural networks (RNN) (i.e., LSTM, GRU), are commonly used to learn features capturing the adjacent structural information for sequence data [13, 28, 37, 38]. While some studies use one type of DL method, some studies create hybrid models by combining different DL methods.

Network-based models have also been explored to analyze sequence data [4, 22]. A common approach for representing genome sequence in a network is to use the De-Bruijn graph [39]. To construct a De-Bruijn graph, the k-mer method is first applied to a sequence input. Each generated k-mer token is used as a node, and subsequent k-mers having an overlap in $k-1$ positions are connected using an edge to construct the graph.

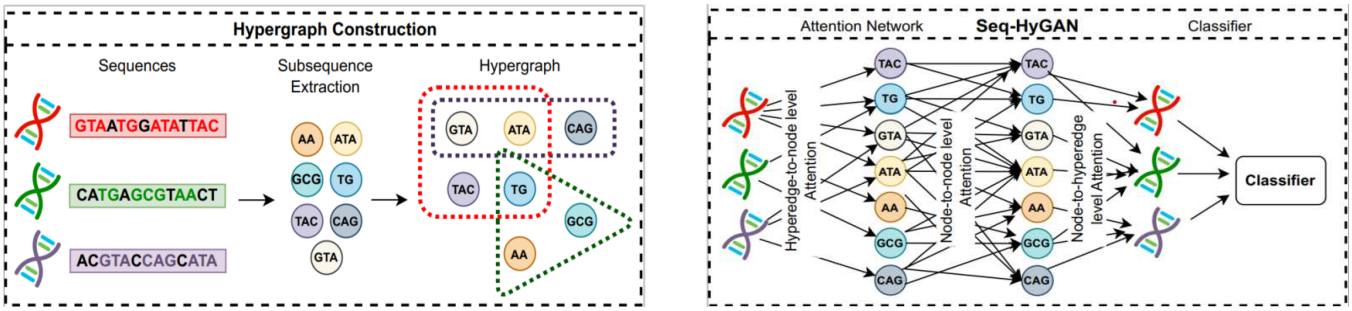
GNNs have exhibited great performances in different research, such as DDI prediction [36], image classification [31], etc. Graph convolutional network (GCN), a popular variant of GNN, has also been applied for sequence data analysis [21, 47]. In [45], authors apply GCN for text classification by creating a heterogeneous text graph including document and word nodes from the whole corpus. The same architecture is applied for DNA-protein binding prediction from sequential data [18]. Their networks have sequence and token nodes extracted from sequences by applying k-mer.

Due to the ability to capture higher-order complex relations, hypergraph and hypergraph neural networks have recently gotten huge attention in different research [14, 24, 33]. These works mainly focus on node representation by using hypergraph neural networks. However, in our proposed Seq-HyGAN, we design a novel three-level attention network to get the representation of hyperedges instead of nodes. Moreover, to the best of our knowledge, this paper is the first to address the sequence classification problem using hypergraph and attention-based hypergraph neural networks.

3 METHODOLOGY

3.1 Preliminaries and settings

Sequence classification is the problem of predicting the class of sequences. Our motivation in this work is that patterns as subsequences are important features of sequences, and if two sequences



(a) Hypergraph Construction (b) Seq-HyGAN

Figure 1: System architecture of the proposed method. The first step is hypergraph construction, where each sequence (e.g., DNA) is a hyperedge, and the (frequent) subsequences of sequences are the nodes. The second step is the Sequence Hypergraph Attention Network, namely Seq-HyGAN, which generates the representations of sequences while giving more attention to the important subsequences and learning the label of the sequences.

share many patterns, they have a higher similarity. Also, it is assumed that similar sequences will have the same class labels. To define the higher-order pattern-based similarity between sequences, we construct a hypergraph from the sequence data. Below is a formal definition of a hypergraph.

Definition 3.1. **Hypergraph:** A hypergraph is defined as $G = (V, E)$ where $V = \{v_1, ..., v_i\}$ is the set of nodes and $E = \{e_1, ..., e_j\}$ is the set of hyperedges. Hyperedge is a special kind of edge that consists of any number of nodes. Similar to the adjacency matrix of a graph, a hypergraph can be denoted by an incidence matrix $H^{n \times m}$ where n is the number of nodes, m is the number of hyperedges and $H_{ij} = 1$ if $v_i \in e_j$, otherwise 0.

After hypergraph creation, to accomplish the sequence classification task, we propose an attention-based hypergraph neural network model consisting of a novel three-level attention mechanism that learns the importance of the subsequences (nodes) and, thus, the representation of the sequences (hyperedges). We train the whole model in a semi-supervised fashion. Our proposed model has two steps: (1) Hypergraph construction from the sequence data, (2) Sequence classification using attention-based hypergraph neural networks.

3.2 Sequence Hypergraph Construction

In order to capture the similarity between sequences, we define the relationship between sequences based on common subsequences within each sequence. We represent this relationship as a hypergraph that captures the higher-order similarity of the sequences. First, we decompose the sequences into a set of subsequences as the important patterns of the sequences. Then, we represent this set of subsequences as the nodes of the hypergraph, and each sequence, including a set of subsequences, is a hyperedge. Each hyperedge may connect with other hyperedges through some shared nodes as subsequences. Thus, this constructed hypergraph defines a higher-level connection of sequences and subsequences and helps to capture the complex similarities between the sequences. Moreover, this hypergraph setting ensures a better robust representation of sequences with a GNN model having a message-passing mechanism not only limited to two nodes but rather between the arbitrary number of nodes and also between edges through nodes. The steps of hypergraph construction are shown in Algorithm 1.

Different algorithms can be used to generate the subsequences, such as ESPF [19], k-mer [46], strobemers [32]. In this paper, we exploit ESPF, and k-mer to generate subsequences and examine their effects on the final results. While k-mer uses all the extracted subsequences for a certain k value, ESPF only selects the most frequent subsequences from a list of candidate subsequences for a certain threshold.

Algorithm 1: Sequence Hypergraph Construction

```
Input: Sequences
Output: Hypergraph incident matrix: H
Subsequence_list<- Sequence_Decomposition(Sequences);
 /* Sequence_Decomposition() could be ESPF or
 k-mer that decomposes sequences into moderated
 size subsequences. */
for each subsequence in Subsequence_list do
    if subsequence is in Sequence_dictionary[sequence] then
        H[i, j] = 1 ; /* i,j is the id of subsequence
          and sequence, respectively. */
    end
end
Output: Hypergraph incident matrix, H
```

ESPF: ESPF stands for Explainable Substructure Partition Fingerprint. As for sub-word mining in the natural language processing domain, ESPF decomposes sequential inputs into a vocabulary list of interpretable moderate-sized subsequences. ESPF considers that a specific sequence property is mainly led by only a limited number of subsequences known as functional groups. Given a database, S of sequences as input, ESPF generates a vocabulary list of subsequences as frequent reoccurring customized size subsequences. Starting with tokens as the initial set, it adds subsequences having a frequency above a threshold in S to the vocabulary list. Subsequences appear in this vocabulary list in order of most frequent to least frequent. In our hypergraph, we use this vocabulary list as nodes and break down any sequence into a series of frequent

subsequences relating to those. For any given sequence as input, we split it in order of frequency, starting from the highest frequency one. An example of splitting a DNA sequence is as follows.

CTGAAAGCAACAGTGAGACGATGAGACCGACGATCCCAGGAGG

⇓

CTGAAAG CAACAG TGAGA CGA TGAGA CCGACGA TCCCAG GAGG

k-mer: k-mer is an effective tool widely used in biological sequence data analysis (e.g., sequence matching). It splits sequential inputs into a list of overlapping subsequence strings of length k. To generate all k-mers from an input string, it starts with the first k characters and then moves by just one character to get the next subsequence, and so on. If t is the length of a sequence, there are $t-k+1$ numbers of k-mers and T^k total possible number of k-mers, where T is the number of monomers. k-mers can be considered as the words of a sentence. Like words, they help to attain the semantic features from a sequence. For example, for a sequence ATGT, monomers: {A, T, and G}, 2-mers: {AT, TG, GT}, 3-mers: {ATG, TGT}.

3.3 Sequence Hypergraph Attention Network

To classify sequences, it is essential to generate feature vectors that can effectively embed the structural information. Since in our hypergraph model, we represent each sequence as a hyperedge; we need to learn hyperedge representation. Regular GNN models that generate the embedding of nodes do not work on our hypergraph. Therefore, we propose a novel Sequence Hypergraph Attention Network model, namely Seq-HyGAN.

Given the f dimensional hyperedge feature matrix, $X \in R^{E \times f}$ and incidence matrix $H \in R^{V \times E}$, Seq-HyGAN generates a hyperedge feature vector of f' dimension via learning a function F. Then it predicts labels for sequences using generated feature vectors.

Our proposed model leverages memory-efficient self-attention mechanisms to capture high-order relationships in the data while preserving both local and global context information. It comprises a three-level attention network: hyperedge-to-node, node-to-node, and node-to-hyperedge levels. At the hyperedge-to-node level, attention is utilized to aggregate hyperedge information and generate node representations that encapsulate global context. The node-to-node level attention refines the node representations by aggregating information from neighboring nodes within the same hyperedge, capturing local context. Lastly, the node-to-hyperedge level attention aggregates node representations from both local and global context perspectives to generate hyperedge representations with attention. We define tree attention layers in general as follows.

$$p_i^l = \mathbf{AG_{E-V}}^l(p_i^{l-1}, n_j^{l-1} | \forall e_j \in E_i), \tag{1}$$

$$m_{i,j}^l = \mathbf{AG_{V-V}}^l(p_i^l, p_y^l | \forall v_y \in e_j), \tag{2}$$

$$n_j^l = \mathbf{AG_{V-E}}^l(n_j^{l-1}, p_i^l, m_{i,j}^l | \forall v_i \in e_j) \tag{3}$$

where $\mathbf{AG_{E\text{-}V}}$ (Hyperedge-to-Node) aggregates the information n_j of all hyperedges e_j to generate the l-th layer representation p_i^l of node v_i and E_i is the set of hyperedges that node v_i belongs to. $\mathbf{AG_{V\text{-}V}}$(Node-to-Node) generate the l-th layer representation $m_{i,j}^l$ of node v_i for a specific hyperedge e_j by aggregating all the nodes v_y present in e_j. Finally, $\mathbf{AG_{V\text{-}E}}$(Node-to-Hyperedge) aggregates the information of all nodes v_i that belongs to hyperedge e_j to generate the l-th layer representation n_j^l of e_j.

Hyperedge-to-node level attention: As our first layer, we learn the representation of nodes via aggregating information from hyperedges to capture the global context in the hypergraph. Although a node may belong to different hyperedges, all hyperedges may not be equally important for that node. To learn the importance of hyperedges for each node and incorporate them into the representation of nodes, we design a self-attention mechanism. While aggregating hyperedge representations for a node, this attention mechanism ensures more weight to important hyperedges than others. With the attention mechanism, the l-th layer node feature p_i^l of node v_i is defined as

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

where α is a nonlinear activation function, W_1 is a trainable weight matrix, and Γ_{ij} is the attention coefficient of hyperedge e_j on node v_i defined as

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

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

where E_i is the set of hyperedges v_i is connected with, β is a LeakyReLU activation function and $*$ is the element-wise multiplication and W_2 and W_3 are trainable weights.

Node-to-node level attention:

The hyperedge-to-node level attention captures global context information while generating node representations. However, while a subsequence is common in the different sequences, they may also have different roles and importance in each sequence. Therefore, it is crucial to capture the local information of nodes specific to hyperedges. Moreover, we need to incorporate the individual contributions of adjacent local subsequences into the representation of a specific subsequence. Furthermore, retaining the positional information of the subsequences in a sequence is essential for the accurate analysis of sequence data. To address these, we introduce a node-to-node attention layer that passes information between nodes in a hyperedge. It learns the importance of subsequences for each other within the same sequence and also incorporates a position encoder that assigns a unique position to each subsequence. To get the position information, we adopt a simple positional encoder inspired by the transformer model [41]. This enables us to preserve local and spatial information of a subsequence within a specific sequence. Using the attention mechanism, the l-th layer node feature $m_{i,j}^l$ of node v_i belonging to hyperedge e_j is defined as

$$m_{i,j}^l = \alpha\left(\sum_{v_y \in e_j} \Phi_{iy} W_4 \bar{p_y}^l\right) \tag{7}$$

$$\bar{p}_y = p_y + \mathrm{PE}(pos_{v_y}) \tag{8}$$

$$\mathrm{PE}(pos, 2x) = \sin(pos/10000^{2x/d}) \tag{9}$$

$$\mathrm{PE}(pos, 2x+1) = \cos(pos/10000^{2x/d}) \tag{10}$$

Where W_4 is a trainable weight, Φ_{iy} is the attention coefficient of neighbor node v_y on node v_i. PE represents the positional encoding function, pos_{v_y} represents the original positional index of v_y in the sequence, $\mathrm{PE}(pos, x)$ refers to the x-th dimension of the positional encoding of the word at position pos in the sequence, and d denotes

the dimension of the positional encoding. The attention coefficient Φ_{iy} is defined as

$$\Phi_{iy} = \frac{\exp(\mathbf{q_y})}{\sum_{q_k \in e_j} \exp(\mathbf{q_k})} \tag{11}$$

$$\mathbf{q_y} = \beta(W_5 \bar{p_y}^l * W_6 \bar{p_i}^l) \tag{12}$$

where e_j is the hyperedge node v_i belongs, W_5 and W_6 are trainable weights.

Node-to-hyperedge level attention: Hyperedge is degree-free that consists of an arbitrary number of nodes. However the contribution of nodes in hyperedge construction may not be the same. To highlight the important nodes for each hyperedge, we employ an attention mechanism. This attention aggregates node representations and assigns higher weights to crucial ones. Moreover, during the aggregation process, it considers the representations of the nodes from both local and global contexts, allowing for a comprehensive understanding of their significance within the hypergraph. With the attention mechanism, the l-th layer hyperedge feature n_j^l of hyperedge e_j is defined as

$$n_j^l = \alpha\left(\sum_{v_i \in e_j} \Delta_{ji} W_7(m_{i,j}^l || p_i^l)\right) \tag{13}$$

where W_7 is a trainable weight, $||$ is the concatenation operation, and Δ_{ji} is the attention coefficient of node v_i in the hyperedge e_j defined as

$$\Delta_{ji} = \frac{\exp(\mathbf{v_i})}{\sum_{v_k \in e_j} \exp(\mathbf{v_k})} \tag{14}$$

$$\mathbf{v_i} = \beta\left(W_8(m_{i,j}^l || p_i^l) * W_9 n_j^{l-1})\right) \tag{15}$$

where v_k is the node that belongs to hyperedge e_j, and W_8, W_9 are trainable weights.

Seq-HyGAN generates the hyperedge representations by employing this three-level of attention. Finally, we linearly project the output of Seq-HyGAN with a trainable weight matrix to generate a C dimensional output for each hyperedge as $Z = nW_c^T$, where C is the number of classes, n is the output of the Seq-HyGAN, and W_c is a trainable weight.

Training: We train our entire model using a cross-entropy loss function defined as

$$L = -\sum_{i=1}^{N}\sum_{c=1}^{C} w_c \log \frac{\exp(Z_{i,c})}{\sum_{j=1}^{C}\exp(Z_{i,j})} y_{i,c} \tag{16}$$

where y is the target, w is the weight, C is the number of classes, and N is the number of samples.

3.4 Complexity

Seq-HyGAN is an efficient model that can be parallelized across the edges and the nodes [42]. Given the f dimensional initial feature of a sequence, it exploits a three-level attention network and generates $f^{'}$ dimensional embedding vector for the sequence. Thus, the time complexity of Seq-HyGAN can be expressed in terms of the cumulative complexity of the attention networks. From equation 4, we can formulate the time complexity for the hyperedge-to-node level attention as: $O(|E|ff^{'} + |V|\kappa f^{'})$, where κ is the average degree of nodes. And in the node-to-node level attention, the time complexity is: $O(|E|(\chi f f^{'} + \chi^2 f^{'}))$, where χ is the average degree of hyperedges. Similarly, we can formulate the time complexity in the node-to-hyperedge level attention as: $O(|V|ff^{'} + |E|\chi f^{'})$.

4 EXPERIMENT

In this section, we perform extensive experiments on four different datasets and five different research problems to evaluate the proposed Seq-HyGAN model. We compute three different accuracy metrics, Precision (P), Recall (R), and F1-score (F1), to analyze and compare our proposed model with the state-of-the-art baseline models. This section starts with a description of our datasets, parameter settings, and baselines, and then we present our experimental results.

4.1 Dataset

We evaluate the performance of our model using four different sequence datasets. They are (1) **Human DNA** sequence, (2) **Anticancer peptides**, (3) **Cov-S-Protein-Seq** and (4) **Bach choral** harmony. All these datasets are publicly available online.

1. The **Human DNA** (HD) sequence dataset consists of 4,380 DNA sequences [9]. Each DNA sequence corresponds to a specific gene family (class), with a total of seven families.

Our objective is to predict the gene family based on the coding sequence of the DNA.

2. The **Anti-cancer peptides** (ACPs) are short bioactive peptides [44]. ACPs are found to interact with vital proteins to inhibit angiogenesis and recruit immune cells to kill cancer cells, such as HNP-110 [20]. These unique advantages make ACPs the most promising anti-cancer candidate [16]. The ACP dataset contains 949 one-letter amino-acid sequences representing peptides and their four different anti-cancer activities (i.e., very active, moderately active, experimental inactive, virtual inactive) on breast and lung cancer cell lines [40]. Given the amino-acid sequence, our goal is to predict the anti-cancer activities.

3. The **Cov-S-Protein-Seq** (CPS) dataset consists of 1,238 spike protein sequences from 67 different coronavirus (CoV) species, including SARS-CoV-2 responsible for the COVID-19 pandemic [3, 6]. The dataset provides information on the CoV species (CVS) and their host species (CHS) [25]. The CoV species are grouped into seven categories, and the host species are grouped into six categories. The goal is to predict the CoV species and host species based on the spike protein sequences.

4. Music is sequences of sounding events. Each event has a specific chord label. The **Bach choral** (BC) harmony dataset consists of 60 chorales containing a total of 5,665 events [40]. Each event is labeled with one of 101 chord labels and described by 14 features. The goal is to predict the chord label based on this information. For the experiment, the five most frequent chord labels out of the 101 chord labels are selected.

4.2 Parameter Settings

We extract subsequences from given sequence datasets using ESPF and k-mer separately to create our hypergraph. With a low-frequency threshold, ESPF produces more subsequences, and all of them may not be important. But with a large-frequency threshold, it produces less number of subsequences, and there might be a missing of some

Table 1: Number of Nodes (N) in the hypergraph based on frequency threshold of ESPF and k value of k-mer

ESPF	HD	BC	ACP	CPS
	$\|N\|$	$\|N\|$	$\|N\|$	$\|N\|$
5	25207	446	382	10776
10	15774	347	225	7987
15	11166	287	166	6769
20	8871	253	138	5971
25	7483	198	110	5477

k-mer	HD	BC	ACP	CPS
	$\|N\|$	$\|N\|$	$\|N\|$	$\|N\|$
5	1247	3220	10301	99794
10	602,855	14462	6799	157,399
15	1,449,240	16163	3103	193,544
20	1,462,963	17909	-	223,073
25	1,467,256	18752	-	248,938

vital subsequences. We choose five different frequency thresholds from 5 to 25 and examine the impact of threshold value on hypergraph learning. Similarly, in k-mer, typically with the increment of k-mer length (i.e., k value), the number of subsequences also increases. We choose five different k values from 5 to 25 and examine their impact on hypergraph learning. However, as Anticancer peptide sequences are too small, we just choose k from 5 to 15. The number of nodes for different threshold values of ESPF and k value of k-mer is given in Table 1 for each dataset.

We perform a random split of our datasets, dividing them into 80% for training, 10% for validation, and 10% for testing. This splitting process is repeated five times, and the average accuracy metrics are calculated and reported in the results section. To find the optimal hyperparameters, a grid search method is used on the validation set. The optimal learning rate is determined to be 0.001, and the optimal dropout rate is found to be 0.3 to prevent overfitting.

We utilize a single-layer `Seq-HyGAN` having a three-level of attention network. one-hot coding is used as an initial feature of the sequences. A LeakyReLU activation function is used on the attention networks side. The model is trained with 1000 epochs and optimized using Adam optimizer. An early stop is used if the validation accuracy does not change for 100 consecutive epochs.

The ML classifiers in subsection: 4.3 are taken from sci-kit learn [29]. For logistic regression (LR), we set the inverse of regularization strength, C=2. A linear kernel with polynomial degree 3 is used in the support vector machine (SVM). Default parameters are used for the decision tree (DT) classifier. The DL models are implemented from Keras layers [10]. In the DL models, RCNN and BiLSTM, we use relu and softmax activation functions in the hidden dense layers and dense output layer, respectively. The models are optimized using Adam optimizer, and dropout layers of 0.3 are used. For node2vec, the walk length, number of walks, and window size are set to 80, 15, and 15, respectively, and for graph2vec, we use the default parameters following the source. We follow DGL [43] to implement graph attention network (GAT). For DNA-GCN and hypergraph neural networks (HGNN, HyperGAT), we use the same hyper-parameters as mentioned in the source papers.

4.3 Baselines

We evaluate our model by comparing its performance with different state-of-the-art models. Based on the data representation style and methodology, we categorize the baseline models into groups below.

1. Machine Learning We utilize CountVectorizer to generate the input features and employ LR, SVM, and DT classifiers. *2. Deep Learning* We use two different hybrid DL models, recurrent convolutional neural networks (RCNN) and bidirectional long short-term memory (BiLSTM), as baselines. *3. Node2vec* We represent each sequence in a graph setting by following a classic method called the De-Bruijn graph as explained in section 2. After constructing the graph, we apply Node2vec [17], which is a random walk-based graph embedding. Node2vec generates the embedding of nodes. To obtain the graph-level representation, we average the embedding of nodes of that graph. Finally, the generated embedding is fed as a feature to the ML classifier for sequence classification. *4. Graph2vec* In the same De-Bruijn graph, we apply the Graph2vec [27] method to generate graph embeddings. Then the graph embeddings are fed into the ML classifiers for sequence classification. *5. Graph Neural Network* Here, we apply graph attention network (GAT) [42] on De-Bruijn graphs to learn the node embedding. Then we get the graph-level embedding using an average pooling-based readout function. Moreover, we follow DNA-GCN [18] to construct a heterogeneous graph from the entire corpus and the extracted subsequences. This graph has two types of nodes: sequence node and subsequence node. After constructing the graph, we apply a two-layer GCN. *6. Hypergraph Neural Network (HNN)* We further compare our model performances with two state-of-the-art hypergraph neural network models: HGNN [14] and HyperGAT [12]. HGNN generates the representation of nodes by aggregating hyperedges. First, we apply HGNN to our hypergraphs and get the node (i.e., subsequence) representations, and then we combine the node representations to get the hyperedge representations. Finally, the hyperedge representations are passed through a classifier. HyperGAT is an attention-based hypergraph neural network that is presented for document classification problems. We apply HyperGAT to our sequence hypergraphs and generate the embeddings of subsequences, and then we apply a mean-pooling layer to get the sequence embedding. Then, the representation is passed through a classifier. In both HGNN and HyperGAT, we use one-hot coding of nodes as initial features.

4.4 Results

4.4.1 Model Performance. We assess the performance of our proposed model by performing extensive experiments on four different datasets for five different problems. For each experiment, we select different threshold values of ESPF and k-mer, and we present the overall performance of our proposed models in terms of the F1-score in Figure 2.

In Figure 2 (a), we depict the performance of our models with a changing frequency threshold of ESPF from 5 to 25. As we can see from this figure, with the increase of ESPF frequency threshold, model performance degrades generally. Especially it shows that the ESPF frequency threshold has a comparatively more significant impact on the Human DNA dataset than others. With the change of frequency threshold from 5 to 25, the F1 score of the Human

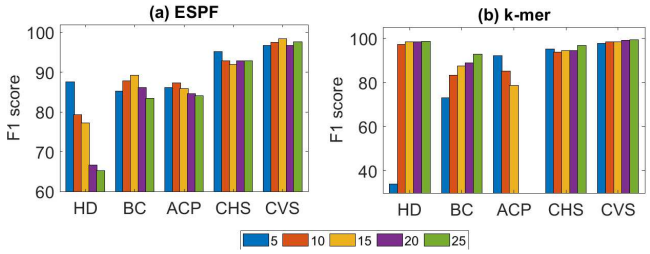


Figure 2: Performance comparison of the proposed model with different thresholds of (a) ESPF and (b) k-mer for different datasets.

DNA dataset has dropped by almost 25%. Since with the increase of frequency threshold, we get a smaller number of subsequences (nodes), and it might not be enough to learn hyperedges with those smaller numbers of nodes (refer Table 1). Generally, frequency threshold 5 yields the best performance for Human DNA and CoV-S-Protein-Seq (for Host species classification) datasets. We get the best performance for Bach choral, Anticancer peptides, and CoV-S-Protein-Seq (for CoV species classification) datasets with frequency thresholds of 15, 10, and 15, respectively.

Figure 2 (b) presents the proposed models' performances for k-mer ranges from 5 to 25. As of ESPF, the effect of parameter k on the results is most for Human DNA than other datasets. When the k value is 5, the F1-score of Human DNA is 33.98%, and for the k value of 25, it is increased by about 190% to 98.83%. The next highest changes are noticed for the Bach choral dataset. With the increase of threshold k from 5 to 25, its F1-score is increased by about 27%. The value of k has comparatively less impact on the results of Cov-S-Protein-Seq datasets. However, interestingly, with the increase of k, the results of the Anticancer peptides dataset decrease. The reasons for these scenarios could be explained by Table 1. In this table, we can see that with the increase in k, the number of nodes in the Human DNA dataset has markedly increased from 1,247 to 1,467,256. This vast number of nodes might capture better information and thus improve the overall performance. On the other hand, for Anticancer peptides, with the increase of k, the number of nodes decreases, hence the graph size, which might degrade the performance.

4.4.2 Comparative Analysis with Baselines. We compare our model results with several state-of-the-art baseline models for each dataset. We consider the thresholds for ESPF and k for k-mer that give the best result for our models for each dataset; as an example, for the Human DNA dataset, we chose $k = 25$ for k-mer that gives the highest score for this data. The experimental results of all models for Human DNA, Bach choral, and Anticancer peptides (Anticancer pept.) datasets are shown in Table 2, and all model results for Cov-S-Protein-Seq dataset for both host species and CoV species are shown in Table 3. In these tables, we can see our models surpass all the baselines thoroughly for all the datasets. More specifically, in Table 2 for the Human DNA dataset, our Seq-HyGAN with k-mer gives the best performance with 98.91%, 98.88%, and 98.83% on precision (P), recall (R), and F1-score (F1). The next best performer is DNA-GCN, which achieves an F1-score of 96.36%, more than 2.5% lower than our model and similar to other accuracy measures. The performances of ML models are also very promising and competitive with our models. All the ML models achieve an F1-score above 80%.

Eventually, for all the datasets, the hypergraph-based model gives the best performance. Specifically, in almost every case, HyperGAT serves as the superior baseline, while HGNN performs as the second-best baseline. The reason for their success lies in their ability to capture higher-order, intricate relationships within a hypergraph structure. Additionally, HyperGAT utilizes attention networks to enhance sequence representation learning, which is superior to HGNN's approach. It is worth mentioning that we apply these models to our hypergraphs, and our experiments demonstrate the effectiveness of representing sequences as hypergraphs.

The overall performance of DNA-GCN is also very competitive with hypergraph-based approaches. DNA-GCN is based on a heterogenous graph that has both sequence and subsequence nodes. It allows information to be passed between the subsequences and also between sequence and subsequence. Moreover, though it does not have any direct connection between sequence nodes, it employs a two-layer GCN that allows the information to be passed between the sequences too. This whole architecture helps it learn a robust representation of sequences. Out of all ML models, generally, SVM generates better output for all the datasets. However, ML models cannot learn features automatically and are limited to external features. In both tables, for all the datasets, the performances of DL models fail to cross the ML classifiers. For example, in the Bach choral dataset, the F1-score of Seq-HyGAN with k-mer is above 93%, and for ML with SVM classifier, it is above 82%. However, the F1-score for RCNN and BiLSTM are 68.23% and 66.32%, respectively. A similar scenario in all other datasets. A possible reason behind the poor performance of DL models could be the size of the data corpus. We know DL models are called data-hungry models. Their performances largely rely on the availability of a bulk amount of label data. However, all our datasets are small.

From Table 2, and 3, we can observe that the performance of Node2vec, Graph2vec, and GAT on the Anticancer peptides dataset is convincing, but their performances on other datasets are abysmal. The network structure of these datasets may be a contributing factor to this discrepancy. To further investigate, we calculate the average network density of each dataset by computing the mean density of its corresponding graphs. For each dataset, the best k-mer is chosen based on the performances of Seq-HyGAN models on that dataset. We find that the Anticancer peptides dataset exhibits the highest average network density of 0.3868, while the Human DNA, Bach Choral, and CoV-S-Protein-Seq dataset graphs have lower densities of 0.0100, 0.2702, and 0.0031, respectively. This disparity in network density could explain the subpar performance of Node2vec, Graph2vec, and GAT on these datasets.

In brief, Seq-HyGAN with k-mer delivers better performances than Seq-HyGAN with ESPF. This may be because ESPF assumes frequent subsequences are the only important ones and eliminates many infrequent subsequences. However, some infrequent subsequences may also be important. Thus, using ESPF, we may seldom lose some infrequent important ones. On the contrary, k-mer does not lose any subsequences; rather, it fetches all and lets the attention model discover the critical ones. In general, a larger k-mer is preferable since it provides greater uniqueness and helps to eliminate the repetitive substrings.

Table 2: Performance comparisons of models for Human DNA, Bach choral and Anticancer datasets

Model	Method	Human DNA			Bach choral			Anticancer pept.		
		P	R	F1	P	R	F1	P	R	F1
ML	LR	92.82	90.64	90.84	88.09	76.68	78.52	77.00	83.16	77.67
	SVM	90.09	85.39	85.83	89.30	80.27	82.08	83.10	86.32	83.27
	DT	92.87	80.37	83.68	86.49	70.85	73.92	78.28	85.32	81.35
DL	RCNN	68.84	37.90	27.86	76.83	71.30	68.23	69.62	80.00	73.34
	BiLSTM	77.80	39.27	35.18	73.93	69.96	66.32	65.70	81.05	72.57
Node2vec	LR	36.09	32.19	22.89	16.11	20.17	18.14	71.32	82.11	75.11
	SVM	10.32	30.82	14.52	14.14	20.18	16.17	66.39	81.05	72.99
	DT	18.04	18.26	18.13	23.03	22.87	22.89	68.75	64.21	66.40
Graph2vec	LR	21.07	26.94	23.63	23.34	19.73	17.81	66.39	81.05	72.99
	SVM	10.32	30.82	14.52	13.09	15.75	16.50	71.32	82.11	75.11
	DT	19.41	19.63	19.39	25.60	25.56	25.48	77.93	73.68	75.54
GNN	DNA-GCN	96.46	96.28	96.36	85.54	85.24	85.27	83.25	83.53	83.82
	GAT	30.06	42.14	36.01	24.75	29.19	31.12	79.23	87.44	79.67
HNN	HGNN	87.03	86.82	87.12	86.12	86.89	86.93	83.82	85.42	83.97
	HyperGAT	85.13	85.33	84.11	88.09	87.44	87.45	85.33	88.42	86.68
Seq-HyGAN	ESPF	88.77	87.89	87.78	89.93	89.72	89.88	91.98	86.75	87.65
	***k*-mer**	**98.91**	**98.88**	**98.83**	**93.78**	**93.10**	**93.18**	**93.36**	**91.72**	**92.33**

Table 3: Performance comparisons of models on Cov-S-Protein-Seq dataset for Host and CoV species prediction

Method	Model	Host species			CoV species		
		P	R	F1	P	R	F1
ML	LR	92.64	91.94	91.48	96.04	95.16	95.21
	SVM	94.20	93.55	93.42	96.60	95.97	96.02
	DT	92.25	91.13	91.25	95.60	94.97	95.02
DL	RCNN	83.22	79.03	76.82	65.53	62.90	57.47
	BiLSTM	70.61	66.94	61.56	66.66	72.58	66.92
Node2vec	LR	26.28	29.03	19.29	12.34	20.16	14.92
	SVM	23.07	24.19	23.27	22.42	25.00	23.54
	DT	20.00	21.77	20.80	23.47	23.39	23.33
Graph2vec	LR	23.74	25.00	23.98	25.92	28.23	26.67
	SVM	17.22	27.42	17.81	17.16	22.58	16.59
	DT	22.68	22.58	22.50	22.80	22.58	22.47
GNN	DNA-GCN	90.91	90.18	91.11	94.34	94.57	94.13
	GAT	22.36	33.23	25.22	24.10	29.65	26.19
HNN	HGNN	91.52	91.91	91.60	94.62	94.42	94.63
	HyperGAT	93.44	93.55	93.14	95.52	95.35	95.45
Seq-HyGAN	ESPF	95.78	95.66	95.49	98.89	98.78	98.72
	***k*-mer**	**97.83**	**96.13**	**97.01**	**99.56**	**99.29**	**99.45**

Moreover, we choose the best-performing method from each baseline model for each dataset; for example, in the case of the Human DNA dataset, we choose LR from the ML models, BiLSTM from the DL models, LR from Graph2vec, DNA-GCN from GNN, HyperGAT from HNN and 25-mer from our Seq-HyGAN models. Then, we compare our models' performances with the baselines by varying the training data sizes from 10% to 80%. A comparison of performance in terms of the F1-score is shown in Fig 3. Results indicate Seq-HyGAN to be the best-performing model, and it still delivers very good results with small training data. However, decreasing the training size affects some baseline models significantly.

The hypergraph's innate ability to capture complex higher-order relationships has made it an effective model for many scientific studies. Seq-HyGAN leverages a hypergraph structure and captures higher-order intricate relations of subsequences within a sequence and between the sequences. Furthermore, it generates a much more robust representation of sequence by utilizing an attention mechanism that discovers the important subsequences of that sequence. While GAT [42] also uses an attention mechanism, it is limited to learning important neighbors of a node and cannot learn significant edges. Additionally, GAT is unsuitable for complex networks with triadic or tetradic relations. The key strength of our proposed model,

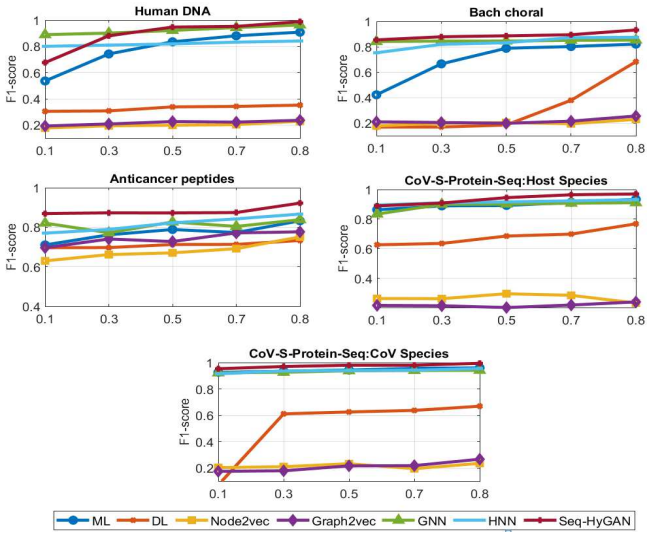


Figure 3: Performance comparison of models for different training sizes

Seq-HyGAN, lies in its three-level attention mechanism, which effectively captures both local and global information. This mechanism allows for the generation of node representations by aggregating information from connected hyperedges (global information) and neighboring nodes (local information) within the same hyperedge, with a specific emphasis on important ones. Likewise, it enables the generation of hyperedge representations by aggregating member nodes, with a particular focus on critical ones.

4.4.3 Space Analysis. Our models demonstrate efficient memory usage by creating only one hypergraph per dataset. Regardless of the chosen thresholds for ESPF and k for k-mer, the number of hyperedges remains consistent across the hypergraphs. In contrast, the De-Bruijn method constructs a separate graph for each sequence, resulting in a significant number of nodes and edges. For instance, in Table 1, we can see that the hypergraph constructed from the Human DNA dataset with a k-mer value of 25 contains 1,467,256 nodes and where the number of hyperedges is the same as the number of sequences in that dataset as mentioned in 4.1 which is 4380. However, in Table 4, we can see that the De-Bruijn graph constructed from the same dataset with the same k-mer value comprises 5,422,447 nodes and 5,418,151 edges. Similar trends are observed in other datasets as well.

4.4.4 Case Study - Impact of hypergraph structure. In contrast to standard graphs, hypergraphs offer the ability to capture higher-order complex relationships that are not easily represented by standard graphs. To demonstrate this capability, we conduct a comparison between our hypergraph-based Seq-HyGAN model and standard graphs. To facilitate this comparison, we construct line graphs from the same datasets, where each sequence is represented as a node, and nodes are connected if they share a certain number (S) of common subsequences (with $S = 2$ in our case). Subsequently, we apply GCN and GAT independently to learn node representations and classify sequences. The performance results are presented in Table 5. The results clearly indicate that our hypergraph-based Seq-HyGAN

Table 4: Number of Nodes (N) and Edges (E) in De-Bruijn Graphs

Dataset	# of nodes	# of edges
	$\|N\|$	$\|E\|$
Human DNA	5,422,447	5,418,151
Bach choral	34111	31890
Anticancer peptides	11939	11058
Cov-S-Protein-Seq	1,576,019	1,574,782

Table 5: F1-score scores for different variants of the model

Dataset	Line graph		Seq-HyGAN	
	GCN	GAT	w/o attn	w/ attn
Human DNA	37.08	39.19	94.13	98.83
Bach coral	75.41	77.65	91.21	93.18
Anticancer peptides	83.52	86.86	89.88	92.33
Host species	72.70	75.31	90.16	97.01
CoV species	80.83	86.50	95.77	99.45

models outperform line graph GCN and GAT models in terms of the F1-score.

4.4.5 Case Study - Impact of Attention Network. In this research paper, we aim to investigate the impact of the attention network in the proposed Seq-HyGAN model on classification performance. To achieve this, we train the model separately with and without the attention network on all datasets and classification problems. The corresponding F1-scores for the test datasets were recorded and are presented in Table 5. The results show that the proposed Seq-HyGAN model with the attention network (w/ attn) outperforms the model without the attention network (w/o attn). Specifically, for CoV-S-Protein-Seq: Host species, the F1-score improved from 90.16% without the attention network to 97.01% with the attention network, representing a significant 7.59% improvement. This improvement can be attributed to the attention network's ability to discover crucial subsequences while learning the sequence representation, which ultimately leads to better performance.

5 CONCLUSION

This paper introduces a novel Sequence Hypergraph Attention Network for sequence classification. Unlike previous models, it leverages a unique hypergraph structure to capture complex higher-order structural similarities among sequences. While sequences are represented as hyperedge, subsequences are represented as nodes in the hypergraph. With the three-level attention model, it learns hyperedge representations as sequences while considering the importance of sequence and subsequences for each other. Our extensive experiments demonstrate that the proposed model surpasses various baseline models in terms of performance. We also show that this model is space and time-efficient with one compact hypergraph setting.

ACKNOWLEDGMENTS

This work has been supported partially by the National Science Foundation under Grant No 2308206.

REFERENCES

[1] Mehmet Emin Aktas and Esra Akbas. 2021. Hypergraph Laplacians in Diffusion Framework. *Studies in Computational Intelligence* 1016 (2 2021), 277–288. https://doi.org/10.48550/arxiv.2102.08867

[2] Mehmet Emin Aktas, Thu Nguyen, Sidra Jawaid, Rakin Riza, and Esra Akbas. 2021. Identifying critical higher-order interactions in complex networks. *Scientific Reports 2021 11:1* 11 (10 2021), 1–11. Issue 1. https://doi.org/10.1038/s41598-021-00017-y

[3] Sarwan Ali, Babatunde Bello, Prakash Chourasia, Ria Thazhe Punathil, Yijing Zhou, and Murray Patterson. 2022. PWM2Vec: An Efficient Embedding Approach for Viral Host Specification from Coronavirus Spike Sequences. *Biology* 11 (3 2022). Issue 3. https://doi.org/10.3390/BIOLOGY11030418

[4] Haitham Ashoor, Xiaowen Chen, Wojciech Rosikiewicz, Jiahui Wang, Albert Cheng, Ping Wang, Yijun Ruan, and Sheng Li. 2020. Graph embedding and unsupervised learning predict genomic sub-compartments from HiC chromatin interaction data. *Nature Communications 2020 11:1* 11 (3 2020), 1–11. Issue 1. https://doi.org/10.1038/s41467-020-14974-x

[5] Holly J Atkinson, John H Morris, Thomas E Ferrin, and Patricia C Babbitt. 2009. Using sequence similarity networks for visualization of relationships across diverse protein superfamilies. *PloS one* 4, 2 (2009), e4345.

[6] Sandrine Belouzard, Jean K. Millet, Beth N. Licitra, and Gary R. Whittaker. 2012. Mechanisms of coronavirus cell entry mediated by the viral spike protein. *Viruses* 4 (2012), 1011–1033. Issue 6. https://doi.org/10.3390/V4061011

[7] Alain Bretto. [n. d.]. Hypergraph Theory. ([n. d.]). http://www.springer.com/series/8445

[8] Bri Bumgardner, Farhan Tanvir, Khaled Mohammed Saifuddin, and Esra Akbas. 2022. Drug-Drug Interaction Prediction: a Purely SMILES Based Approach. (1 2022), 5571–5579. https://doi.org/10.1109/BIGDATA52589.2021.9671766

[9] Nagesh Singh Chauhan. [n. d.]. Demystify DNA Sequencing with Machine Learning | Kaggle. https://www.kaggle.com/code/nageshsingh/demystify-dna-sequencing-with-machine-learning/notebook

[10] Francois Chollet et al. 2015. *Keras*. https://github.com/fchollet/keras

[11] Phillip EC Compeau, Pavel A Pevzner, and Glenn Tesler. 2011. Why are de Bruijn graphs useful for genome assembly? *Nature biotechnology* 29, 11 (2011), 987.

[12] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be more with less: Hypergraph attention networks for inductive text classification. *arXiv preprint arXiv:2011.00387* (2020).

[13] Jesse Eickholt and Jianlin Cheng. 2013. DNdisorder: Predicting protein disorder using boosting and deep networks. *BMC Bioinformatics* 14 (3 2013), 1–10. Issue 1. https://doi.org/10.1186/1471-2105-14-88/FIGURES/6

[14] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2018. 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* (9 2018), 3558–3565. https://doi.org/10.48550/arxiv.1809.09401

[15] Vladimir Gligorijević, P. Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C. Taylor, Ian M. Fisk, Hera Vlamakis, Ramnik J. Xavier, Rob Knight, Kyunghyun Cho, and Richard Bonneau. 2021. Structure-based protein function prediction using graph convolutional networks. *Nature Communications 2021 12:1* 12 (5 2021), 1–14. Issue 1. https://doi.org/10.1038/s41467-021-23303-9

[16] Francesca Grisoni, Claudia S Neuhaus, Miyabi Hishinuma, Gisela Gabernet, Jan A Hiss, Masaaki Kotera, and Gisbert Schneider. [n. d.]. De novo design of anticancer peptides by ensemble artificial neural networks. ([n. d.]). https://doi.org/10.1007/s00894-019-4007-6

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[18] Yuhang Guo, Xiao Luo, Liang Chen, and Minghua Deng. [n. d.]. DNA-GCN: Graph convolutional networks for predicting DNA-protein binding. ([n. d.]). https://github.com/Tinard/dnagcn.

[19] Kexin Huang, Cao Xiao, Lucas Glass, and Jimeng Sun. 2019. Explainable substructure partition fingerprint for protein, drug, and more. In *NeurIPS Learning Meaningful Representation of Life Workshop.*

[20] Kai Yao Huang, Yi Jhan Tseng, Hui Ju Kao, Chia Hung Chen, Hsiao Hsiang Yang, and Shun Long Weng. 2021. Identification of subtypes of anticancer peptides based on sequential features and physicochemical properties. *Scientific Reports 2021 11:1* 11 (6 2021), 1–13. Issue 1. https://doi.org/10.1038/s41598-021-93124-9

[21] Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. [n. d.]. Text Level Graph Neural Network for Text Classification. ([n. d.]). https://www.cs.umb.edu/

[22] Sohyun Hwang, Chan Yeong Kim, Sunmo Yang, Eiru Kim, Traver Hart, Edward M. Marcotte, and Insuk Lee. 2019. HumanNet v2: human gene networks for disease research. *Nucleic acids research* 47 (1 2019), D573–D580. Issue D1. https://doi.org/10.1093/NAR/GKY1126

[23] Muhammad Ifte Islam, Farhan Tanvir, Ginger Johnson, Esra Akbas, and Mehmet Emin Aktas. 2021. Proximity-based compression for network embedding. *Frontiers in big Data* 3 (2021), 608043.

[24] Eun-Sol Kim, † Woo, Young Kang, Kyoung-Woon On, Yu-Jung Heo, Byoung-Tak Zhang, and Kakao Brain. [n. d.]. Hypergraph Attention Networks for Multimodal Learning. ([n. d.]). https://spacy.io/

[25] Kiril Kuzmin, Ayotomiwa Ezekiel Adeniyi, Arthur Kevin DaSouza, Deuk Lim, Huyen Nguyen, Nuria Ramirez Molina, Lanqiao Xiong, Irene T. Weber, and Robert W. Harrison. 2020. Machine learning methods accurately predict host specificity of coronaviruses based on spike sequences alone. *Biochemical and Biophysical Research Communications* 533 (12 2020), 553–558. Issue 3. https://doi.org/10.1016/J.BBRC.2020.09.010

[26] Christina Leslie, Eleazar Eskin, and William Stafford Noble. 2001. The spectrum kernel: A string kernel for SVM protein classification. In *Biocomputing 2002*. World Scientific, 564–575.

[27] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).

[28] Ngoc Giang Nguyen, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen Rosyking Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, Kenji Satou, Ngoc Giang Nguyen, Vu Anh Tran, Duc Luu Ngo, Dau Phan, Favorisen Rosyking Lumbanraja, Mohammad Reza Faisal, Bahriddin Abapihi, Mamoru Kubo, and Kenji Satou. 2016. DNA Sequence Classification by Convolutional Neural Network. *Journal of Biomedical Science and Engineering* 9 (4 2016), 280–286. Issue 5. https://doi.org/10.4236/JBISE.2016.95021

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[30] Yujie. Qian. 2019. A graph-based framework for information extraction. (2019). https://dspace.mit.edu/handle/1721.1/122765

[31] Alyssa Quek, Zhiyong Wang, Jian Zhang, and Dagan Feng. 2011. Structural image classification with graph neural networks. *Proceedings - 2011 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2011* (2011), 416–421. https://doi.org/10.1109/DICTA.2011.77

[32] Kristoffer Sahlin. 2021. Strobemers: an alternative to k-mers for sequence comparison. *bioRxiv* (2021), 2021–01.

[33] Khaled Mohammed Saifuddin, Bri Bumgardnerr, Farhan Tanvir, and Esra Akbas. 2022. HyGNN: Drug-Drug Interaction Prediction via Hypergraph Neural Network. *arXiv preprint arXiv:2206.12747* (2022).

[34] Khaled Mohammed Saifuddin, Muhammad Ifte Khairul Islam, and Esra Akbas. 2021. Drug Abuse Detection in Twitter-sphere: Graph-Based Approach. *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021* (2021), 4136–4145. https://doi.org/10.1109/BIGDATA52589.2021.9671532

[35] Hiroto Saigo, Jean Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. 2004. Protein homology detection using string alignment kernels. *Bioinformatics (Oxford, England)* 20 (7 2004), 1682–1689. Issue 11. https://doi.org/10.1093/BIOINFORMATICS/BTH141

[36] Farhan Tanvir, Khaled Mohammed Saifuddin, and Esra Akbas. 2022. DDI Prediction via Heterogeneous Graph Attention Networks. (7 2022). https://doi.org/10.48550/arxiv.2207.05672

[37] Patrick Brendan Timmons and Chandralal M. Hewage. 2021. ENNAACT is a novel tool which employs neural networks for anticancer activity classification for therapeutic peptides. *Biomedicine & pharmacotherapy = Biomedecine & pharmacotherapie* 133 (1 2021). https://doi.org/10.1016/J.BIOPHA.2020.111051

[38] Mirko Torrisi, Gianluca Pollastri, and Quan Le. 2020. Deep learning methods in protein structure prediction. *Computational and Structural Biotechnology Journal* 18 (1 2020), 1301–1310. https://doi.org/10.1016/J.CSBJ.2019.12.011

[39] Isaac Turner, Kiran V Garimella, Zamin Iqbal, and Gil Mcvean. [n. d.]. Integrating long-range connectivity information into de Bruijn graphs. ([n. d.]). https://doi.org/10.1093/bioinformatics/bty157

[40] UCI. [n. d.]. UCI Machine Learning Repository: Data Sets. https://archive.ics.uci.edu/ml/datasets.php?format=&task=&att=&area=&numAtt=&numIns=&type=seq&sort=nameUp&view=table

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[42] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[43] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. (9 2019). https://doi.org/10.48550/arxiv.1909.01315

[44] Mingfeng Xie, Dijia Liu, and Yufeng Yang. 2020. Anti-cancer peptides: classification, mechanism of action, reconstruction and modification. *Open Biology* 10 (7 2020). Issue 7. https://doi.org/10.1098/RSOB.200004

[45] Liang Yao, Chengsheng Mao, and Yuan Luo. [n. d.]. Graph Convolutional Networks for Text Classification. ([n. d.]). www.aaai.org

[46] Jingsong Zhang, Jianmei Guo, Xiaoqing Yu, Xiangtian Yu, Weifeng Guo, Tao Zeng, and Luonan Chen. 2017. Mining k-mers of various lengths in biological sequences. In *Bioinformatics Research and Applications: 13th International Symposium, ISBRA 2017, Honolulu, HI, USA, May 29–June 2, 2017, Proceedings 13*. Springer, 186–195.

[47] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. [n. d.]. Every Document Owns Its Structure: Inductive Text Classification via Graph Neural Networks. ([n. d.]). https://github.com/CRIPAC-DIG/TextING

[48] Jaroslaw Zola. 2014. Constructing similarity graphs from large-scale biological sequence collections. In *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*. IEEE, 500–507.