IMPROVING REPRESENTATION LEARNING IN GRAPH NEURAL NETWORKS

by

Tuğrul Hasan Karabulut

B.S., Mathematical Engineering, Yıldız Technical University, 2021

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2025

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Assist. Prof. İnci M. Baytaş, for her exceptional mentorship, insightful guidance, and consistent encouragement throughout this incredibly challenging journey.

I would also like to extend my sincere appreciation to my jury members, Prof. Lale Akarun and Assoc. Prof. Yusuf Yaslan for accepting to participate in the defense of my thesis and for their valuable feedback.

I would like to thank Assist. Prof. Suzan Üsküdarlı and Gökçe Uludoğan for valuable research discussions and brainstorming sessions in our study group, which enhanced my research perspective throughout this thesis study.

ABSTRACT

IMPROVING REPRESENTATION LEARNING IN GRAPH NEURAL NETWORKS

Real-world data often comprises interconnected entities rather than isolated objects. Traditional neural networks struggle to process irregular structures inherent in prominent domains such as social networks, bioinformatics, and physical systems that have extensively benefited from deep learning. Graph neural networks (GNNs) following the message-passing paradigm have emerged as a versatile and effective framework for processing relational data with arbitrary structural characteristics. However, the efficiency of GNNs comes with challenges in representation learning that impact downstream task performance. This thesis addresses two major challenges occurring in deep GNNs: over-smoothing and over-squashing. The message-passing mechanism by design is prone to produce almost identical representations after a certain number of steps, causing an over-smoothing effect. We propose an adaptive channel-wise message-passing approach to alleviate over-smoothing. The proposed model, Channel-Attentive GNN (CHAT-GNN), learns how to attend to neighboring nodes and their feature channels. Thus, more diverse information can be transferred between nodes during message-passing. Since information between distant nodes is transmitted through sequential message-passing between adjacent nodes, community-connecting nodes must compress substantial information into low-dimensional feature spaces, resulting in the over-squashing effect. We propose incorporating local virtual nodes that increase the feature capacity of central nodes and the number of paths through central regions. Moreover, our shared virtual node embedding approach enables communication with range greater than the receptive field of GNNs. We compare both approaches with state-of-the-art baselines in terms of downstream task performance and conduct detailed experiments to validate the effectiveness of CHAT-GNN and local virtual nodes.

ÖZET

ÇİZGE SİNİR AĞLARINDA TEMSİL ÖĞRENİMİNİ GELİŞTİRMEK

Gerçek hayattaki veriler birbirinden ayrı nesnelerden ziyade birbirine bağlı varlıklardan oluşur. Geleneksel sinir ağları, derin öğrenmeden çok fazla fayda sağlayan sosyal ağlar, biyoinformatik ve fiziksel sistemler gibi alanlarda bulunan düzensiz yapıları işlemekte zorlanırlar. Mesaj iletimi paradigmasını kullanan Çizge Sinir Ağları (GNN), farklı yapıdaki topolojiler içeren ilişkisel verileri işlemek için esnek ve etkili bir çerçeve olarak ortaya çıkmıştır. Ancak, GNN'lerin verimliliği, temsil öğrenmede yaşanan ve görev performansını oldukça etkileyen zorluklarla birlikte gelir. Bu tezde çizge modellerinde ortaya çıkan iki büyük zorluğu ele alıyoruz: aşırı-yumuşatma ve aşırı-sıkıştırma. Mesaj iletim mekanizması, tasarımı gereği belirli bir adım sayısından sonra neredeyse özdeş temsiller üretme eğilimindedir, bu da aşırı-yumuşatma etkisine neden olur. Aşırı-yumuşatmayı hafifletmek için uyarlanabilir, kanal bazlı bir mesaj iletim yaklaşımı öneriyoruz. Önerilen model, Kanal-Dikkatli GNN (CHAT-GNN), komşu düğümlere ve özellik kanallarına nasıl dikkat edeceğini öğrenir. Böylece, mesaj iletimi sırasında düğümler arasında daha çeşitli bilgiler aktarılabilir. Ayrıca, uzak düğümler komşu düğümler arasındaki mesaj iletimi yoluyla iletişim kurduğundan, çizgedeki toplulukları bağlayan düğümlerin düşük boyutlu özellik uzayında büyük miktarda bilgi depolaması gerekir, bu da aşırı-sıkıştırma etkisine neden olur. Bu tezde merkezi düğümlerin özellik uzayı boyutluluğunu artırırken, aynı zamanda merkezi bölgeler üzerinden geçen yol sayısını da artıran sanal düğümler eklemeyi öneriyoruz. Paylaşımlı sanal düğüm gömme yaklaşımımız, GNN'in algı alanından daha uzun menzilli bir iletişim sağlar. Her iki yaklaşımımızı da çeşitli problemler üzerindeki performansları açısından literatürdeki en başarılı yöntemlerle karşılaştırıyoruz. Ek olarak, CHAT-GNN ve yerel sanal düğümlerin verimliliğini doğrulamak için ayrıntılı deneyler gerçekleştiriyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbf{A}$	Adjacency matrix
D	Number of hidden layer dimensions
F	Number of input features
G	Graph
K	Number of classes
$\mathcal{N}$	Neighborhood function
x	Scalar
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
σ	Sigmoid function
ϕ	Linear transformation function
$\odot$	Element-wise multiplication operator

LIST OF ACRONYMS/ABBREVIATIONS

CHAT-GNN	Channel-Attentive Graph Neural Network
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GIN	Graph Isomorphism Network
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
LVN	Local Virtual Nodes

1. INTRODUCTION

Graph machine learning has a growing popularity due to the advances in representation learning with deep neural networks. Many applications in prominent domains, especially complex physical systems [1], recommender systems [2, 3], natural language processing [4, 5], computer vision [6, 7], and protein bioinformatics [8, 9], leverage deep neural networks for learning high-quality representations of graph structures. Graph Neural Network (GNN) is the main architectural framework specialized in efficiently generating representation from graphs in an end-to-end fashion [10–13].

GNNs comprise of a family of neural network architectures operating on graphs. Most GNNs conform to the message-passing paradigm [10, 11, 13, 14]. The message-passing framework is based on the idea of applying local operations, i.e. between nodes, to reach a global solution, thereby paving the way to process of arbitrarily large graphs with many nodes and edges [14–16]. GNNs are able to encode information from further away nodes into the node's representation by the means of exchanging and aggregating messages among the neighboring nodes. Graph convolution is one instantiation of message-passing. Graph Convolutional Networks (GCNs) perform the first-order approximation of spectral graph convolution, equivalent to aggregating the neighbor representations scaled by node degrees [10, 12]. Graph Attention Networks (GAT) employs attention mechanism that learns the importance of each neighbor to perform a weighted aggregation during message-passing [11, 17]. In addition, GraphSAGE further improves the efficiency of message-passing by neighborhood sampling [13].

Number of message-passing steps in a GNN determines a node's receptive field. GNNs with many layers incorporates information from far away nodes into node's representation. Similarly to neural networks benefiting from many layers [18, 19], it is also crucial to design GNN architectures that can scale to many message-passing layers. Message-passing algorithm is remarkably efficient in tasks that largely depends on locality by design. However, if the problem requires communication between nodes at a

distance that exceeds locality, an ill-formed message-passing algorithm could be incapable of solving this long-range task [20–25]. Various phenomena that needs addressing occur when message-passing is scaled to large number of layers. In this thesis, we focus on two significant issues in modern GNNs: *over-smoothing* and *over-squashing*.

Over-smoothing issue in GNNs refers to the gradual decrease in the distinguishability of node representations across message-passing layers [20–23]. Li *et al.* showed that GCNs apply a form of Laplacian smoothing on graphs and the produced node representations converge to an identical representation for nodes in the same component [20]. Using small number of layers could prevent the over-smoothing issue and could be sufficient for local tasks. However, tasks that need deeper propagation requires special treatment. Guided by this perspective, there has been ongoing research to alleviate over-smoothing in recent years [23, 26–28]. Attention mechanism is promising since it can be used to select which information to aggregate during message-passing and potentially leading to more high-fidelity representations. In recent years, many attention-based GNNs that perform well on various benchmarks are proposed [11, 17, 27, 29–31]. In GNNs, self-attention mechanism can be used to assign importance scores to neighbors [11, 17, 27, 31] or the messages from individual hops [27, 29, 30] which we indicate as edge attention and hop attetion, respectively [27]. Edge attention helps in diversifying the messages between nodes that have common neighbors. However, attention-based aggregation is fundamentally a weighted sum of neighbor representations and thus prone to over-smoothing [11, 17, 27]. Hop attention helps with selecting the information from earlier layers that generate less smoothed features while reducing the effect of later layers. However, the hop attention is not involved in message-passing, the main source of over-smoothing.

To achieve more adaptive aggregation during message-passing, we propose a channel-wise attention mechanism that can send specialized messages based on each edge and feature channel. Our architecture, called CHAT-GNN [32], which employs channel-wise attention, performs well on various node classification benchmarks and can be scaled to many layers without suffering from over-smoothing [32].

On the other hand, over-squashing is another phenomenon in GNNs arising from topological deficiencies in the input graphs, such as bottlenecks [24]. Although GNNs favor locality, many tasks, e.g. molecular property prediction [33], require long-range interactions between nodes. However, each successive layer in GNNs causes an exponential increase in the information to be encoded in node representations [24]. However, it is not feasible to store long-range information in fixed-size representations that are determined prior to training. Bottleneck regions further amplify this catastrophe since the information exchange between distant nodes should be made through a few nodes. As a result, message-passing operating on fixed-size node representation is prone to over-squashing of information, which prevents effective learning on graphs. There have been many studies in recent years that aim to combat over-squashing from topology and model perspectives. Topological methods apply, namely *graph rewiring* that alters the graph topology by adding edges, simplifying the information propagation during message-passing, and optionally removing redundant edges to reduce the computational overhead. Graph rewiring could help in both eliminating the bottlenecks of the graph by creating more pathways for the information to be propagated. Topological methods can be further classified into two types: spectral and spatial methods. Spectral graph rewiring methods rewire the graph to optimize a metric based on the Laplacian matrix of the graph while spatial rewiring is used to improve the local connectivity of the graph. Although graph rewiring helps to mitigate the structural issues in graphs, it disrupts the valuable domain knowledge represented in the graph structure. Further, finding the bottlenecks in graphs are computationally infeasible in large graphs as they require eigendecomposition of graph Laplacian or graph curvature measures. In addition, it is not clear how to incorporate graph rewiring in some tasks such as edge-level tasks, since rewiring introduces new edges to the graph while also removing existing ones.

Besides modifying the graph structure with rewiring, over-squashing can also be mitigated with sufficient GNN width, as this increases the representation capacity of the network. Di Giovanni *et al.* [25] theoretically validate this notion by defining a influence measure based on the norm of the Jacobian of node representations from

the input and the output layer and proving that its upper bound scales with the width of the GNN. However, having a neural network with a large width requires more computation and prone to over-fitting [25]. Therefore, it is promising to investigate how we can systematically increase the representation capacity of a graph to combat over-squashing, a subject that, to the best of our knowledge, has received limited attention to date.

We propose *Local Virtual Nodes*, which feature shared trainable embeddings that enhance the graph's knowledge capacity without distorting its structure. These nodes can be seamlessly integrated into any GNN architecture. By strategically positioning these virtual nodes to support central nodes, our approach achieves two key benefits: it creates additional pathways for information flow from dense regions to sparse regions, while also enabling feature representations that operate independently of the GNN's standard receptive field through the shared trainable embeddings. Our method outperforms well-known graph rewiring methods in the literature by a large margin. Additional experiments further show our method mitigates over-squashing.

To summarize, in this thesis, we tackle two prominent issues in graph representation learning with message-passing. To tackle over-smoothing, we propose a channel-wise, or feature-wise message-passing mechanism that can adapt different aggregation schemes based on different edges and feature channels. We design a GNN architecture called CHAT-GNN [32] that uses the aforementioned channel-attentive mechanism. Experiments on well-known node classification benchmark datasets show that CHAT-GNN surpasses the performance of baseline methods on many of the heterophilous graphs. In addition, we train CHAT-GNN with many message-passing layers and observe no performance drop which indicates resilience to over-smoothing. As for over-squashing, we propose Local Virtual Nodes that enlarge the feature capacity of graphs with negligible computational cost. Further, shared trainable virtual node embeddings can be used to learn structure-informed features which transcend receptive field and can be reached without long-range communication. Our key contributions in this thesis include:

- We propose a channel-wise graph attention mechanism that is more adaptive than traditional graph attention and enables representation diversity between neighboring nodes.
- Channel-attentive message-passing layer is designed to be integrated into any GNN architecture indepedendent of the input graph and the task.
- The visualization of the attention weights shows that the channel-wise attention of CHAT-GNN can, in fact, adapt to different neighbors and hops.
- We propose the notion of Local Virtual Nodes for expanding the representation capacity of a graph's central regions.
- We suggest that equipping the group of virtual nodes assigned to each central node to have shared learnable features to improve generalization and communicating long-range information.

2. RELATED WORK

We provide a review of the GNN literature, beginning with message-passing GNNs and their variations. Then, we cover the challenges in representation learning with GNNs and existing approaches addressing them.

2.1. Message-Passing GNNs

Message-passing is an effective approach to learn representations on graphs [14, 16]. Gilmer *et al.* proposed a framework describing a T-step procedure to obtain node-level and graph-level representations. Each message-passing step has a message function, M_t, that outputs the message to be communicated given the adjacent node representations, and an update function, U_t, that produces the final feature vector based on the aggregated messages [14]. An optional readout function, R, can be incorporated after the message-passing steps to obtain a graph-level representation. Battaglia *et al.* further generalized this framework with a GNN block updating edge, node and graph representations simultaneously [16].

There have been many proposed GNN architectures that are inspired by the message-passing framework or can be categorized within this paradigm [10, 11, 13]. GCN applies message-passing using a degree-normalized adjacency matrix which acts as an approximated spectral filter [10]. GAT aggregates messages from neighbors via attention-based averaging [11]. GraphSAGE improves the efficiency of message-passing with neighborhood sampling and separately transforms the node and the neighborhood representations [13]. Graph Isomorphism Network (GIN) extends the expressivity of message-passing to match the Weisfeiler-Lehman (WL) graph isomorphism test [34].

2.2. Attention & Gating

Attention-based GNNs [11, 17, 27, 31, 35] have a growing popularity in various graph-based tasks. Graph attention applies the well-known attention mechanism [36–38] over the neighborhood of a node to obtain an edge-specific weighting that enables more expressive message aggregation instead of the uniform aggregation in standard GCN. GAT is the seminal work for the use of attention in message-passing [11]. Brody *et al.* showed that GAT, in fact, applies *static attention*, meaning that ranking of attention scores are same across all source nodes [17]. Further, they proposed a modified graph attention called, GATv2, that applies *dynamic attention* [17]. FAGCN enables both positive and negative attention scores to enhance the flexibility of aggregation and tackle graphs with high heterophily [31]. Song *et al.* further investigate the graph attention methods in the literature from over-smoothing perspective and propose an improved model called AERO-GNN [27]. Moreover, some studies propose attending over the message-passing layers so that each layer gets a different weight [27, 29, 30]. This type of attention is referred to as hop attention [27]. In our approach, CHAT-GNN, extends graph attention by learning a channel-wise weighting mechanism that determines the rate of the information flow for each channel in an edge.

The proposed framework is also closely related to the gating mechanism in neural networks. The recurrent neural networks utilize the gating mechanism to control the information flow between consecutive time steps [39, 40]. In the context of GNNs, Li *et al.* proposed using GRUs to update node representations at each message-passing step [41]. Further, Bresson and Laurent introduced a GNN with edge-based gating to solve subgraph matching and clustering [42]. FAGCN [31] employs a self-gating mechanism that adjusts the frequency of the underlying filtering in message-passing. On the other hand, Gradient gating [28] uses graph gradients to control the information flow in a multi-rate fashion. Ordered GNN [43] aims to order the information coming from different hops by considering the message-passing scheme for a node as a rooted tree growing with the hops. The model uses a gating mechanism to separate the information from different hops in the message vector. Thus, the primary motivation

is to prevent the mixing of node features from different hops [43].

The CHAT-GNN has similarities and differences between the standard attention and gating mechanisms. The proposed channel-wise attention aims to control high and low-frequency changes in the feature channels while learning to attend to each neighboring node, layer, and feature channel differently. Thus, the proposed message-passing framework extends the graph attention and gate ideas to each direction in the GNN's latent space.

2.3. Learning on Heterophilous Graphs

Node classification on heterophilous graphs is widely studied [29,43–45]. A graph is considered heterophilous if most of its edges are between the nodes from different classes. Learning on heterophilous graphs is challenging since the standard message-passing mechanisms used in GCN, GAT, and similar models enforce the representation of adjacent nodes to be close to each other. Consequently, the discriminability of the node representations is sensitive to message-passing operations in heterophilous graphs.

Some studies on heterophilous graphs focus on handling neighborhood orders differently by learning hop weights [27,29] and separating the representation of different hops [44]. Separating the node's representation and its neighborhood representation during the combine phase of message-passing is also emphasized by some studies [44,45]. The separation of representations prevents us from mixing the two representations that are supposed to be dissimilar in heterophilous graphs [44]. The GraphSAGE [13] model also has a separate embedding mechanism for the node and its neighborhood. Platonov *et al.* showed that GraphSAGE [13] performs better than GCN and GAT in heterophilous graph datasets proposed by the authors [45]. We also follow this principle by applying separate learnable affine transformations to the node embeddings and their neighborhood representation. Similarly to prior work, we also observe that allowing this mixing does not hurt the performance in graphs with high homophily.

2.4. Over-smoothing

Over-smoothing is an issue in GNNs that results in indistinguishable node representations when stacking many message-passing layers [20, 21, 23]. This issue stems from the message-passing layers acting as low-pass filters and over-smoothing the feature vectors of closer nodes [20, 22, 23]. As a result, adding layers to GNNs with such message-passing mechanisms induces underfitting. However, graphs can have arbitrary topological structures, and the task of interest may require leveraging the underlying deep and complex connections in the graph. Many studies aim to mitigate the over-smoothing issue from various perspectives, such as data augmentation and graph topology optimization [23, 46]. For instance, Rong *et al.* proposed randomly removing some edges during training to prevent overfitting and over-smoothing [46]. On the other hand, Chen *et al.* proposed topological metrics to quantify over-smoothing and a framework that optimizes the proposed metrics [23]. They came up with a metric called MADGap that measures the difference between the mean average distance of remote and neighboring nodes. Further, they propose to use a regularizer term improving MADGap and an algorithm an adaptive edge optimization algorithm removing inter-class edges and adding intra-class edges [23].

More recent studies focus on the connection between over-smoothing and the information flow during message passing, and thus, they address over-smoothing with gating [28, 31, 43], and attention [27, 29, 30]. Bo et al. demonstrate that their FAGCN model uses a gating mechanism with negative values to combine both low-frequency and high-frequency information during aggregation, producing more discriminative representations for neighboring nodes and thus reducing over-smoothing [31]. Similary, gradient gating GNN (G^2GNN) employs a channel-wise gating mechanism as an adaptive residual connection between the input and the output of the message-passing layer [28]. Rusch *et al.* proved that gradient gating prevents over-smoothing by providing a formal analysis of GNNs as discrete dynamical systems [28]. Song *et al.* developed a GNN with an *ordered gating* mechanism that encodes the features from different hops in distinct dimensions of node representations. Several other approaches uti-

lize attention mechanisms to mitigate over-smoothing, including Deep Adaptive GNN (DAGNN), Generalized PageRank GNN (GPR-GNN), and Attentive Deep Propagation GNN (AERO-GNN), each implementing distinct attention strategies to preserve node distinctiveness across multiple layers. DAGNN learns node-specific, positive hop attention weights based on the aggregated features from multiple hops [30], while GPR-GNN learns global hop attention weights that allow both negative and positive values [29]. Both approaches helps with over-smoothing by selecting information from specific hops [27,29,30]. Lee *et al.* proposed AERO-GNN, that combines both edge attention and hop attention, and analytically compared its resistance to over-smoothing against other attention-based GNNs [27]. and systematically compared its resistance to over-smoothing against other attention-based GNNs. However, these existing approaches employ only a single attention (or gating) weight during message-passing. Our channel-wise attention mechanism provides greater flexibility during message aggregation, enabling the combination of both low and high-frequency information across different channels.

The analyses in this thesis show that the performance of the proposed CHAT-GNN framework is less vulnerable than the baselines against deepening the GNN.

2.5. Over-squashing

Over-squashing refers to the phenomenon of long-range information being squashed into feature vectors of limited capacity [24]. The over-squashing issue is mainly caused by bottlenecks in graphs, which are subgraphs with relatively few nodes that are critical connections between dense regions. Various methods for structural modifications have been proposed in recent years to overcome over-squashing. Alon and Yahav [24] suggested treating the graph a as a complete graph in the last layer of GNN lowers the graph's diameter to two and solve the bottleneck issue. They show that this approach performs better than using the raw input graph for all layers. Likewise, graph transformers are also said to be resillient to over-squashing since they globally aggregate information from all nodes using the attention mechanism and feed

the structural information into the network with positional encodings [47–49]. However, both approaches have quadratic complexity in terms of the number of nodes, as opposed to linear complexity of message-passing GNNs, and are infeasible to use in scenarios with scarce resources and large graphs. Consequently, modifying the graph structure while leveraging traditional GNNs emerges as a favorable approach to mitigate over-squashing as well as improving the connectivity of the graph in general.

Most graph rewiring approaches function as preprocessing steps before training, while in recent years dynamic rewiring algorithms have also been developed. Diffusion Improves Graph Learning (DIGL) learns a graph diffusion matrix by combining power of the adjacency matrix and feeds the sparsified diffusion matrix as the input graph to the GNN [50]. Stochastic Discrete Ricci Flow (SDRF), introduced by Topping *et al.* , uses a metric called Ricci curvature that quantifies to what extent an edge acts as a bottleneck [51]. New edges are added around the endpoints of an edge with low Ricci curvature and edges with high Ricci curvature are removed [51]. Batch Ollivier-Ricci Flow (BORF) is another curvature-based rewiring approach that tackles both over-smoothing and over-squashing [52]. Black *et al.* [53] leverages total effective resistance as a measure of connectivity of a graph. Their algorithm, called Greedy Total Resistance (GTR) rewiring, rewires the graph in a way that optimizes total effective resistance. Greedy Local Edge Flip (G-RLEF) similarly rewires the graph by sampling node pairs based on effective resistance and adding edges around the sampled node pairs [54]. First-order Spectral Rewiring (FoSR) adds edges to the graph that optimizes the spectral gap the most [55]. Further, the authors propose using relational GNNs that handle the original and new edges differently [55]. Gabrielsson *et al.* suggest that applying local rewiring to connect nodes within a certain distance combined with graph positional encoding achieves significant performance improvement over baseline GNNs and graph transformer [56]. Locality-Aware Sequential Rewiring (LASER) is a more systematic local rewiring method that preserve both locality and sparsity of the graph [57]. Some studies proposed dynamically learning how to rewire the graph rather than as a preprocessing algorithm [58, 59]. DiffWire is a dynamic rewiring approach that estimates commute time and spectral gap of a graph with a neural

network, and then rewires the graph based on the estimated values [58]. Qian *et al.* proposed a probabilistic rewiring framework that first generate scores for each edge, then applies a differentiable sampling that selects edges to construct the rewired graph [59]. Finally, another line of work is utilizing graph types that known to have well connectivity properties and are void of bottlenecks. These methods propose converting to given graph to a specific type of graph that a message-passing GNNs may work well. Expander graphs, Cayley graphs and Delaunay graphs are utilized to avoid structural issues in the original graph [60–62].

Virtual nodes are studied as an alternative that balances the message-passing and global aggregation approaches such as graph transformers. A virtual node is originally defined as a *supernode* that is connected to all other nodes [14,16]. Cai *et al.* [63] proved that a message-passing GNN with a virtual node and large enough width can simulate transformers. Further, Southern *et al.* [64] showed that a global virtual node can help reduce the overall commute time, a metric that is computed using the spectrum of the graph. The virtual node idea is also explored in various tasks [65–67]. VN-EGNN extends EGNN [68] with a set of global virtual nodes per graph to solve protein binding site identification task [66]. Concurrently, Zhang *et al.* employed global virtual nodes to improve learning on large geometric graphs [67]. Hwang *et al.* proposed incorporating virtual nodes for the link prediction task [65]. They include virtual nodes into the graph by clustering the graph and assigning each cluster a virtual node and connecting it to all nodes in the cluster [65]. However, they do not focus on mitigating over-squashing but rather they analyze the expressiveness under the WL-test. We extend the virtual node idea by introducing strictly local virtual nodes. Moreover, we view virtual nodes as a means to increase the representation capacity of a central node rather than a solely new node added to the graph. Finally, our globally shared learnable virtual node embeddings further serve as a tool to extend the receptive field of GNNs.

Finally, inspired by the theoretical relationship between the GNN width and over-squashing [25], Choi *et al.* [69] proposed expanding the width of central nodes and applying different message-passing function between of from different types. Our

approach is similar in the sense that virtual node group assigned to a central node serve as hubs for the original node and implicitly expands the width for that specific node.

3. METHOD

We start by presenting some basic definitions related to graphs and GNNs, which are necessary to formulate our proposed methods. The following sections start with the input representation and the message-passing mechanism used in GNNs.

3.1. Input Representation

We are interested in undirected graphs without self-loops and edge weights. Such graphs can be denoted as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. We denote the set $\{0, 1, 2, \ldots, N\}$ as $[N]$. $G[S]$ represents the induced subgraph of G for the set $S \subseteq \mathcal{V}$, which includes all the edges betweeen nodes in S. Let N be the number of nodes in the graph. Alternatively, we can also represent a graph by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. $\mathbf{D}$ is a diagonal matrix in which each entry in the diagonal represents the node's degree. We also denote the neighbor set of a node $v \in \mathcal{V}$ by $\mathcal{N}(v)$. Each node has an input feature vector $\mathbf{x} \in \mathbb{R}^F$, and all of them together form an input feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$, where F is the number of input features. We denote the model's hidden dimension as D (also referred to as channels) and represent the l-th hidden layer's output for node v as $\mathbf{h}_v^{(l)} \in \mathbb{R}^D$. For the node classification task is considered, each node must have a label $\mathbf{y} \in \{0, 1\}^K$, where K is the number of available classes in the dataset. For the graph classification task, we consider a dataset defined as $\mathcal{D} = \{(G_1, y_1), (G_2, y_2), \ldots, (G_n, y_n)\}$ with n graphs along with their corresponding labels.

3.2. Graph Neural Networks

Most GNNs utilize the message-passing paradigm due to its simple local operations to calculate representations on graphs efficiently [14, 16], and it achieves strong performance on various transductive and inductive tasks [10, 11, 13, 14]. The basic idea of message-passing is calculating node representations by receiving "messages"

from immediate neighbors, aggregating them into a single representation of the whole neighborhood, and repeating this process as many times as required to allow information to flow from distant nodes. Assume that $\mathbf{h}_v^{(k)} \in \mathbb{R}^D$ is the representation of the node v at layer k. The message-passing mechanism in GNNs for the layer $k+1$ can be formalized as

$$\mathbf{m}_v^{(k+1)} = \mathrm{AGG}^{(k+1)}\left(\left\{\mathrm{MSG}^{(k+1)}(\mathbf{h}_v^{(k)}, \mathbf{h}_w^{(k)}) \mid w \in \mathcal{N}(v)\right\}\right) \tag{3.1}$$

$$\mathbf{h}_v^{(k+1)} = \mathrm{CMB}^{(k+1)}\left(\mathbf{h}_v^{(k)}, \mathbf{m}_v^{(k+1)}\right) \tag{3.2}$$

where $\mathrm{AGG}^{(k+1)}$ is any permutation-invariant aggregation function, $MSG^{(k+1)}$ is the message function calculating the message that will passed from node w to node v, and finally, $\mathrm{CMB}^{(k+1)}$ is the combine function that combines the node's current representation and the neighborhood representation, $\mathbf{m}_v^{(k+1)}$, into a single representation, $\mathbf{h}_v^{(k+1)}$, as the updated representation of node v at layer $k+1$. The design choices for these three fundamental functions distinguish the message-passing GNNs in the literature [10, 11, 13, 17, 31].

3.3. Channel-Attentive Graph Neural Networks

In this thesis, we are specifically interested in the $\mathrm{MSG}(\cdot,\cdot)$ function since it has been shown that designing message functions that can learn to output specialized messages between different nodes helps in various tasks [11, 17, 27, 31]. For example, GCN uses the following message function:

$$\mathrm{MSG}(\mathbf{h}_v, \mathbf{h}_w) = \frac{\mathbf{h}_w}{\sqrt{d_v d_w}} \tag{3.3}$$

where d_v is the degree of node v. While this message function is simple yet achieves competitive results on some datasets [10], attention-based [11, 17, 27, 31] GNNs use

more expressive message functions that learn a weighting between pairs of nodes as

$$\text{MSG}(\mathbf{h}_v, \mathbf{h}_w) = \alpha(\mathbf{h}_v, \mathbf{h}_w)\mathbf{h}_w \tag{3.4}$$

where $\alpha : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is the weighting function that outputs a scalar, called attention coefficients or attention scores indicating the importance of the neighbor w to the node v. The function $\alpha(\cdot, \cdot)$ is often designed using fully connected layers [11, 17, 27, 31]. Note that we assume unnormalized attention functions to indicate that the output depends on $\mathbf{h}_v$ and $\mathbf{h}_w$. These attention weights can be normalized over the node's neighborhood using an activation function, such as softmax, so that the sum of the scores always equals one [11, 17]. The following section explains the proposed channel-wise attention for the neighboring nodes.

3.3.1. Channel-Attentive Message-Passing

Although traditional attention-based message-passing performs well on some benchmarks, it still suffers from over-smoothing and fails on many heterophilous graphs. Let $\alpha_{vw} = \alpha(\mathbf{h}_v, \mathbf{h}_w)$ be the attention score between a pair of adjacent nodes v and w. The fact that each feature channel gets multiplied by the same attention score, α_{vw}, and the distinction between messages passed to the arbitrary neighbors w and u being simply defined with the ratio α_{vw}/α_{vu} prevent the model from learning distinctive representations of each neighbor during aggregation. In addition, we may want to diminish the contribution of a feature from a specific neighbor and increase the effect of that feature from another neighbor. Further, we may want to control the rate of information flow from each feature channel just like the traditional attention-based models controlling the contribution of each neighbor. With this motivation, we consider the weighting functions of the form $\beta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^D$. We denote $\beta_{vw} = \beta(\mathbf{h}_v, \mathbf{h}_w)$ as the attention weight vector between the pair of nodes v and w, and β_{vwd} denotes the importance of the feature channel d for the specific node pair v and w. Accordingly,

we reformulate our message function to use the channel-wise importance as

$$\text{MSG}(\mathbf{h}_v, \mathbf{h}_w) = \beta(\mathbf{h}_v, \mathbf{h}_w) \odot \mathbf{h}_w \tag{3.5}$$

where $\odot$ denotes the element-wise multiplication, also known as the Hadamard product.

The channel-attentive function β of CHAT-GNN is designed as a fully connected module optimized during training in an end-to-end fashion. The β_{vwd} values are crucial for the expressivity of the message-passing phase. Unlike the standard attention with output values between zero and one, we allow negative values to capture negative interactions between the neighboring nodes that help mitigate over-smoothing [31, 70]. Therefore, the proposed channel-attentive function is computed as [32]

$$\beta(\mathbf{h}_v, \mathbf{h}_w) = \tanh\left(\mathbf{W}_1\mathbf{h}_v + \mathbf{W}_2\mathbf{h}_w\right) \tag{3.6}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{D\times D}$ are trainable weight matrices multiplied by the source and the neighbor nodes, respectively. The contribution of allowing negative values can be explained by an analogy with the filtering in image processing. A low-pass filter applied to local regions on an image has a smoothing effect that reduces the noise in the image. While a high-pass filter emphasizes high-frequency regions, such as edges, resulting in a sharpening effect. Similarly, we employ the message-passing mechanism that acts on local regions of the graph, which are neighborhoods of nodes. Both filtering approaches could be desirable in an arbitrary graph structure, especially in large and complex real-world graphs. The proposed message-passing formulation resembles a hybrid low and high-pass filtering applied on the same region but on different color channels.

The full expression of the aggregation part of the channel-attentive message-passing mechanism can be formalized as

$$\mathbf{m}_v = \sum_{w\in\mathcal{N}(v)} \frac{1}{\sqrt{d_v d_w}} \tanh(\mathbf{W}_1\mathbf{h}_v + \mathbf{W}_2\mathbf{h}_w) \odot \mathbf{h}_w \tag{3.7}$$

where $\mathbf{m}_v$ is the neighborhood representation obtained by the channel-attentive message-passing mechanism. Note that we use the summation as the aggregation operator AGG in Equation (3.1) and the inverse square root of degrees for scaling [10, 31]. The equation above shows that each feature channel is attended differently, and the attention weights are computed considering the neighboring features. Thus, the network can adjust the channel-wise message-passing mechanism by informing each feature channel about other feature channels.

3.3.2. The Combine Phase

We consider the following expression to combine message-passing for an arbitrary layer $k+1$:

$$\mathbf{h}_v^{(k+1)} = \phi_1^{(k+1)}(\mathbf{h}_v^{(k)}) + \phi_2^{(k+1)}(\mathbf{m}_v^{(k+1)}) \tag{3.8}$$

where $\phi_1 : \mathbb{R}^D \to \mathbb{R}^D$ and $\phi_2 : \mathbb{R}^D \to \mathbb{R}^D$ are linear transformations applied on the current node representation and the neighborhood representation, respectively. As hinted in the literature [13, 44, 45], separately projecting the node and neighborhood representations performs better than directly mixing the two representations. Furthermore, Zhu *et al.* [44] showed that, in some cases, this approach results in a more capable model for heterophilous settings. In our experiments, we observe that adding separate transformations, ϕ_1, ϕ_2, improves the performance in some graphs in both high and low homophily settings. However, we achieve the best results for most datasets without separate linear projection layers. Therefore, we treat this choice as a hyperparameter to tune between the identity function $\phi_1(\mathbf{x}) = \mathbf{x}$ and a single linear layer $\phi_1(\mathbf{x}) = \mathbf{W}_1'\mathbf{x}$ discussed with an ablation study in Subsection 4.4.6.

3.3.3. CHAT-GNN: The Whole Architecture

Now, we present the proposed CHAT-GNN [32] architecture illustrated in Figure 3.1. Let us consider the forward propagation for an arbitrary node v. Assume that the input features of node v are denoted as $\mathbf{x}_v$. First, the input node features are

projected into a hidden representation with a non-linear layer as

$$\mathbf{h}_v^{(0)} = \text{ReLU}\left(\mathbf{W}_{\text{in}}\mathbf{x}_v + \mathbf{b}_{\text{in}}\right) \tag{3.9}$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{D\times F}$, $\mathbf{b}_{\text{in}} \in \mathbb{R}^{D\times 1}$ are weight and bias of the initial feature map layer. We denote $\mathbf{h}_v^{(0)}$ as the initial input to the message-passing layers. The initial feature projection is beneficial to reducing the large number of sparse input features and learning fine-grained features that will be passed to the message-passing mechanism. Following the footsteps of previous studies [26, 31, 45], we observed that applying residual connection between the initial features and the message-passing layers helps speed up the training process since the model may want to access the initial fine-grained features in deeper message-passing layers.

Now, suppose that we are at the $(k+1)$-th message-passing layer. A message-passing layer in CHAT-GNN operates as

$$\mathbf{m}_v^{(k+1)} = \sum_{w\in\mathcal{N}(v)} \frac{1}{\sqrt{d_v d_w}} \tanh\left(\mathbf{W}_1^{(k+1)}\mathbf{h}_v^{(k)} + \mathbf{W}_2^{(k+1)}\mathbf{h}_w^{(k)}\right) \odot \mathbf{h}_w^{(k)} \tag{3.10}$$

$$\tilde{\mathbf{h}}_v^{(k+1)} = \phi_1^{(k+1)}(\mathbf{h}_v^{(k)}) + \phi_2^{(k+1)}(\mathbf{m}_v^{(k+1)}) \tag{3.11}$$

where $\phi_1^{(k+1)}$ and $\phi_2^{(k+1)}$ are the linear projection layers at the layer $k+1$. Finally, we apply the initial residual connection and the layer normalization on the updated features as

$$\mathbf{h}_v^{(k+1)} = \text{LayerNorm}^{(k+1)}\left(\tilde{\mathbf{h}}_v^{(k+1)} + \mathbf{h}_v^{(0)}\right) \tag{3.12}$$

where $\text{LayerNorm}(\cdot)^{(k+1)}$ is the layer normalization at the $(k+1)$-th message-passing layer, applied on the feature channels [71]. Stacking many message-passing layers increases the norm of node representations that cause instabilities and overfitting. Therefore, GNNs benefit from various normalization techniques that help tackle over-smoothing [72, 73]. For instance, AERO-GNN uses a predetermined series of scalars to scale the output representations [27]. On the other hand, Platonov *et al.* [45] used layer normalization and residual connections to boost the performance of GCN, GAT,

and GraphSAGE on heterophilous graphs. We also observe the benefit of layer normalization, especially in deeper architectures, for large enough heterophilous graphs. For extremely small graphs and graphs with high homophily, the overfitting problem occurs since the neighborhood patterns can easily be memorized.

Finally, an output projection layer is applied to the final node representations to generate the prediction for the task at hand as

$$\mathbf{z}_v = \mathbf{W}_{\text{out}}\mathbf{h}_v^{(L)} + \mathbf{b}_{\text{out}} \tag{3.13}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{K\times D}$, $\mathbf{b}_{\text{out}} \in \mathbb{R}^{K\times 1}$ are the weight and bias of the output layer and L is the number message-passing layers in CHAT-GNN. The proposed architecture is trained for the node classification task. Therefore, the softmax function is used as the output layer's activation. The cross-entropy loss function given below is used to optimize the proposed and baseline models:

$$L = -\frac{1}{N_{\text{train}}} \sum_{n=1}^{N_{\text{train}}} \sum_{k=1}^{K} y_{nk} \cdot \log(\hat{y}_{nk}) \tag{3.14}$$

where N_{train} denotes the number of training nodes, K denotes the number of classes, and $\hat{y}_{nk}$ and y_{nk} are the predicted and true labels of node n, respectively.

In summary, an end-to-end GNN architecture coined CHAT-GNN is proposed to learn adaptive weights from layer, neighbor, and feature channels with a channel-attentive message-passing mechanism. The following sections present discussions on differences between CHAT-GNN and multi-head attention, followed by a theoretical anaylsis outlining the effectiveness of CHAT-GNN in preventing over-smoothing.

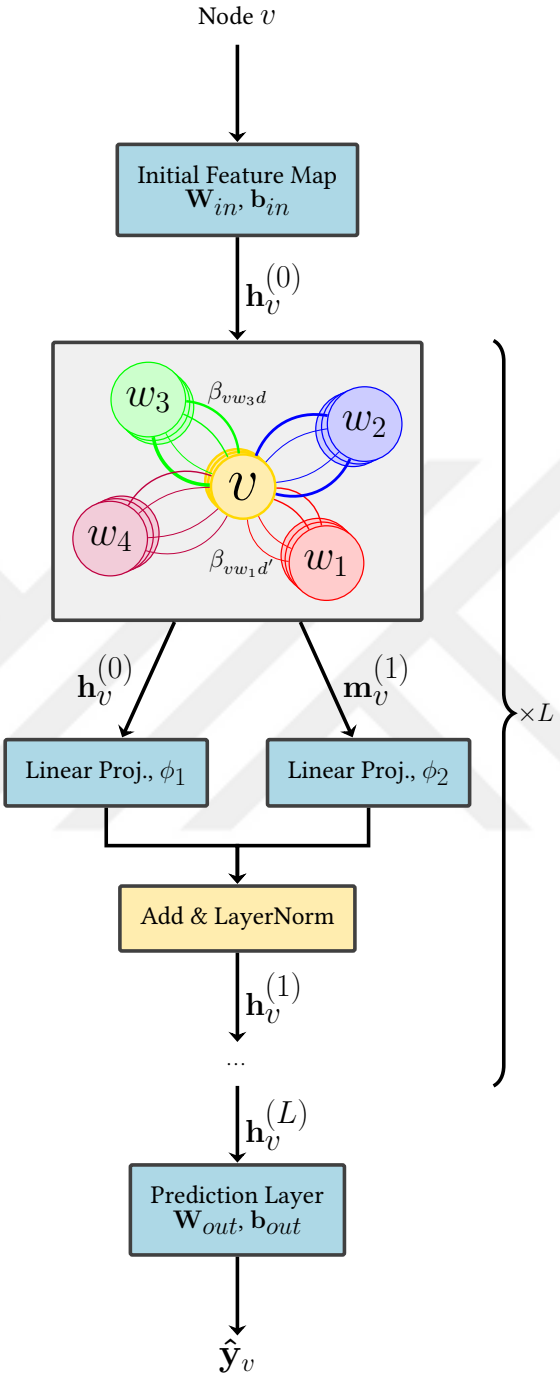


Figure 3.1. The architecture of CHAT-GNN. After a shared input layer projects the node features, L channel-attentive message-passing layers are included. Each message-passing operation is followed by linear projections and layer normalization. The final prediction is obtained by feeding the output of the L-th layer to an output layer.

3.3.4. Comparison with Multi-Head Attention

As discussed in previous sections, our approach is essentially an extension of standard attention to each feature channel. Multi-head attention might seem to lie between these two ideas. Multi-head attention maps the input representations to different subspaces to learn more diverse output representations [11,38]. The learned representation in each head is concatenated to generate the final output. However, it still applies the standard attention mechanism in each head. For example, multi-head graph attention with K heads is illustrated as [11]

$$\mathbf{m}_v = \Big\Vert_{k=1}^{K} \sigma\big(\sum_{w \in \mathcal{N}(v)} \alpha_{vwk} \mathbf{W}_k \mathbf{h}_w \big) \tag{3.15}$$

where α_{vwk} is the attention score between nodes v and w for attention head k, and $\mathbf{W}_k$ is the weight matrix that extracts features for head k to attend on.

On the other hand, in channel-wise attention, as formalized in Equation (3.7), we calculate channel-wise importance by considering the original node representations, and all feature channels are able to interact with each other. Also, the learned attention scores are used to weigh the input representations. Therefore, channel-wise attention differs from other attention-based mechanisms, offering a more flexible information flow during message-passing.

3.3.5. Theoretical Analysis

In this section, we discuss the effectiveness of channel-attentive message-passing in reducing over-smoothing by providing some theoretical findings. We start with the notion of local smoothness. *Local variation* [74] of node v according to graph signal $\mathbf{X}$ is defined as

$$E_v(\mathbf{X}) = \sum_{w \in \mathcal{N}(v)} \|\mathbf{x}_v - \mathbf{x}_w\|^2. \tag{3.16}$$

Then, the *Dirichlet Energy* is defined as the sum of local variations across all nodes, which can be seen as a measure of global smoothness [74, 75] and denoted as

$$E(\mathbf{X}) = \frac{1}{N} \sum_{v \in \mathcal{V}} E_v(\mathbf{X}). \tag{3.17}$$

Thus, the global smoothness of the node features depends on the local variations. The following proposition indicates that the change in the local variation between consecutive GNN layers is bounded by the distance between the aggregated messages.

Proposition 3.1. *Let us assume that we have a GNN in which the (k+1)-th layer's update is as follows:* $\mathbf{h}_v^{(k+1)} = \mathbf{h}_v^{(k)} + \mathbf{m}_v^{(k+1)}$*, where* $\mathbf{m}_v^{(k+1)}$ *is the output of a message-passing layer. Then, the change in the local variation of node* v*,* $\Delta_{(k+1)} E_v = E_v^{(k+1)} - E_v^{(k)}$ *satisfies the following inequality:*

$$\Delta_{(k+1)} E_v \leq \sum_{w \in \mathcal{N}(v)} (\delta_{wv}^{(k+1)})^2 + 2c\delta_{wv}^{(k+1)} \tag{3.18}$$

where $\delta_{wv}^{(k+1)} = \|\mathbf{m}_w^{(k+1)} - \mathbf{m}_v^{(k+1)}\|$ *and* $c = \max \|\mathbf{h}_w^{(k)} - \mathbf{h}_v^{(k)}\|$.

Proof.

$$\begin{aligned}
\Delta_{(k+1)} E_v &= \sum_{w \in \mathcal{N}(v)} \left(\|\mathbf{h}_w^{(k)} + \mathbf{m}_w^{(k+1)} - (\mathbf{h}_v^{(k)} + \mathbf{m}_v^{(k+1)})\|^2 - \|\mathbf{h}_w^{(k)} - \mathbf{h}_v^{(k)}\|^2 \right) \\
&= \sum_{w \in \mathcal{N}(v)} \Big[\left(\|\mathbf{h}_w^{(k)} + \mathbf{m}_w^{(k+1)} - (\mathbf{h}_v^{(k)} + \mathbf{m}_v^{(k+1)})\| - \|\mathbf{h}_w^{(k)} - \mathbf{h}_v^{(k)}\| \right) \\
&\quad \times \left(\|\mathbf{h}_w^{(k)} + \mathbf{m}_w^{(k+1)} - (\mathbf{h}_v^{(k)} + \mathbf{m}_v^{(k+1)})\| + \|\mathbf{h}_w^{(k)} - \mathbf{h}_v^{(k)}\| \right) \Big]
\end{aligned} \tag{3.19}$$

Using the triangle inequality, we get:

$$\Delta_{(k+1)} E_v \leq \sum_{w \in \mathcal{N}(v)} \Big[\left(\|\mathbf{m}_w^{(k+1)} - \mathbf{m}_v^{(k+1)}\| \right) \left(\|\mathbf{m}_w^{(k+1)} - \mathbf{m}_v^{(k+1)}\| + 2\|\mathbf{h}_w^{(k)} - \mathbf{h}_v^{(k)}\| \right) \Big]$$

$$= \sum_{w \in \mathcal{N}(v)} (\delta_{wv}^{(k+1)})^2 + 2c\delta_{wv}^{(k+1)} \tag{3.20}$$

□

Proposition 3.1 shows that the upper bound on the change in the local variation is determined by how different the message vectors of neighboring nodes are. The over-smoothing effect begins when a GNN layer's output for a node is similar to its neighbors. Next, we present another proposition that compares the standard graph attention and our proposed channel-wise graph attention in terms of how likely the $\delta_{wv} = \|\mathbf{m}_w - \mathbf{m}_v\|$ would be non-zero.

Proposition 3.2. *Assume that we have a pair of nodes v and w that satisfy $\mathcal{N}(v) = \mathcal{N}(w)$, and positive node features $\mathbf{X}$. Further, assume that message vectors with standard and channel-wise attention are calculated as $\mathbf{m}_v = \sum_{k \in \mathcal{N}(v)} \alpha_{vk}\mathbf{x}_k$ and $\mathbf{m}'_v = \sum_{k \in \mathcal{N}(v)} \beta_{vk} \odot \mathbf{x}_k$, respectively. Lastly, we assume that each unnormalized attention score α_{vk} and β_{vkd} are all independent random variables. Then, we have the following inequalities satisfied:*

$$\Pr(\delta^*_{m_{vw}} > 0) \leq |\mathcal{N}(v)|P^* \tag{3.21}$$

$$\Pr(\delta^*_{m'_{vw}} > 0) \leq |\mathcal{N}(v)|DP^* \tag{3.22}$$

where $\delta^*_{m_{vw}} = \sum_{k \in \mathcal{N}(v)} |\alpha_{vk} - \alpha_{wk}|\|\mathbf{x}_k\|$ and $\delta^*_{m'_{vw}} = \sum_{k \in \mathcal{N}(v)} \|(\beta_{vk} - \beta_{wk}) \odot \mathbf{x}_k\|$ are the respective upper bounds of δ_{vw} under the standard and channel-wise attention settings, D is the dimensionality of feature vectors and channel-wise attention vector β, and P^* is the maximum possible value that $\Pr(|\alpha - \alpha'| > 0)$ can attain for two arbitrary attention scores α and α'.

Proof. We begin with the first inequality. We can see that $\delta^*_{m_{vw}}$ is greater than zero if for at least one neighbor k, we have $|\alpha_{vk} - \alpha_{wk}| \neq 0$. Therefore, using Boole's inequality,

we have the following:

$$\begin{aligned}\Pr(\delta^*_{m_{vw}} > 0) &= \Pr(\bigcup_{k\in\mathcal{N}(v)} \{|\alpha_{vk} - \alpha_{wk}| \neq 0\}) \\ &\leq \sum_{k\in\mathcal{N}(v)} \Pr(|\alpha_{vk} - \alpha_{wk}| \neq 0) \\ &\leq |\mathcal{N}(v)|P^* \end{aligned} \tag{3.23}$$

Now, we prove the second equality. Similarly, we see that the $\delta^*_{m'_{vw}}$ is greater than zero if for at least one dimension d for one neighbor k, we have $|\beta_{vkd} - \beta_{wkd}| \neq 0$. Again, we utilize Boole's inequality as follows:

$$\begin{aligned}\Pr(\delta^*_{m'_{vw}} > 0) &= \Pr(\bigcup_{k\in\mathcal{N}(v)} \{\|\beta_{vk} - \beta_{wk}\| \neq 0\}) \\ &\leq \sum_{k\in\mathcal{N}(v)} \Pr(\|\beta_{vk} - \beta_{wk}\| \neq 0) \\ &= \sum_{k\in\mathcal{N}(v)} \Pr(\bigcup_{d=1}^{D} \{|\beta_{vkd} - \beta_{wkd}| \neq 0\}) \\ &\leq \sum_{k\in\mathcal{N}(v)} \sum_{d=1}^{D} \Pr(|\beta_{vkd} - \beta_{wkd}| \neq 0) \\ &\leq |\mathcal{N}(v)|DP^* \end{aligned} \tag{3.24}$$

□

Therefore, in the worst case where two nodes v and w have exactly the same neighborhood, the channel-attentive attention mechanism is more likely to avoid collapsed message vectors. Next, we describe our local virtual nodes framework to address over-squashing, another major challenge in GNNs that this thesis investigates.

3.4. Local Virtual Nodes

In this section, we introduce local virtual nodes. First, we start with our motivation for this approach. Subsequently, we formulate our method while also comparing it with graph rewiring techniques and other virtual node methods.

3.4.1. Motivation

Over-squashing is an issue in which the model is incapable to store abundant information in a limited feature space. Over-squashing can be attributed either to the topology of the graph or the width of the GNN [25]. Increasing width may help for graphs where the structural issues are not severely pathological and the problem radius is low. However, excessive increase in width may bring about challenges in generalization [25]. Consequently, graph rewiring have received more attention in the literature as a convenient preprocessing step, allowing standard GNNs to be implemented without modification. Graph rewiring techniques help mitigate over-squashing by connecting distant nodes with direct edges, which would otherwise be required to communicate through bottleneck pathways. Di Giovanni *et al.* provides a theoretical framework proving both approaches could help in combatting over-squashing [25]. The authors utilize the Jacobian of a GNN's output layer's output with respect to the first layer's input:

$$\left\| \frac{\partial \mathbf{h}_v^{(l)}}{\partial \mathbf{h}_u^{(0)}} \right\|_1 \leq \underbrace{(c_\sigma w p)^l}_{\text{model}} \underbrace{(\mathbf{S}^l)_{vu}}_{\text{topology}} \tag{3.25}$$

where c_σ is the Lipschitz constant of the activation function σ, w is the maximum value of all the weight matrices, p is the model's width, and $\mathbf{S}$ is the graph shift operator defined based on the adjacency matrix, $\mathbf{A}$ and used for message-passing. For example, GCN's graph shift operator is $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.

The Equation (3.25) (Theorem 3.2 in [25]) demonstrates that the sensitivity of a GNN's output with respect to its input is dependent on both the model parameters, such as the activation function and network width, as well as underlying graph

topology. c_σ is derived from the common activation function choices (ReLU, sigmoid etc.) and the weights are determined during training and are usually kept from exploding with regularization. Thus, the network width p and the graph shift operator $\mathbf{S}$, which depends on the topology, remain as the configurable parameters to combat over-squashing.

3.4.2. Graph Rewiring Methods

Since over-squashing was initially analyzed from a topological perspective and mainly characterized by bottleneck structures [24], graph rewiring techniques have been more extensively adopted. Graph rewiring enables the decoupling of the computation graph from the raw input graph, which is a topic studied in different contexts including over-smoothing and expressitivity [46, 76]. Given a graph $G = (V, E)$, graph rewiring can be formulated as

$$\mathcal{R}(G) = G_{\text{RW}} = (V, E \cup E_+ \setminus E_-) \tag{3.26}$$

where E_+ and E_- denote the set of edges that are added and removed, respectively. Most graph rewiring techniques are applied prior to training, whereby the rewired graph G_{RW} is fed into GNN instead of the original graph G. Many rewiring methods build upon the original set of edges and introduce new ones that improve connectivity [51,53–55, 57]. Besides preprocessing, iterative graph rewiring [57] and dynamic rewiring [58, 59] have also been explored. Another category of rewiring is completely disregarding the original edges and form a new set of edges to construct a well-known graph (Expander, Cayley, Delaunay, etc.) that have an optimal topology for feature propagation [60–62]. Despite their demonstrated efficacy in mitigating over-squashing, we identify the following limitations of graph rewiring approaches:

- "Artifical" edges introduced to the graph lack semantic meaning, thereby breaking the inductive bias inherently enforced by the input graph.
- Similarly, removing existing edges may lead to poorer input representation. Further, it is not clear how to apply graph rewiring with edge removal on edge-level tasks such as edge-property prediction.

- New edges may lead to over-smoothing [52].
- Methods relying on curvature measures or eigendecomposition to pinpoint bottlenecks introduce computational complexity [51–54].

3.4.3. Global Virtual Nodes

Global virtual nodes constitute an alternative methodology that bridges the global aggregation mechanism of graph transformers with the localized message-passing paradigm of GNNs. [63, 64]. A global virtual node help in reducing the diameter of the graph to two and introducing an additional set of features storing global information. Therefore, we avoid quadratic complexity of graph transformers, or using complete graphs, while still propagating global information through the global virtual node. Similarly to graph rewiring, augmenting a graph with a global virtual node can be formulated as

$$\mathrm{VN}(G) = G_{\mathrm{VN}} = (\mathcal{V} \cup \{v_{N+1}\}, \mathcal{E} \cup \{(v_j, v_{N+1}) \mid j \in [N]\}) \tag{3.27}$$

where v_{N+1} denotes the global virtual node. A single virtual node may not be able to capture all global (or long-range) information. Sestak *et al.* introduces multiple global virtual nodes per graph with learnable features to attenuate over-squashing for binding site prediction task [66]. Consequently, we can parameterize the global virtual function $\mathrm{VN}(\cdot)$ as

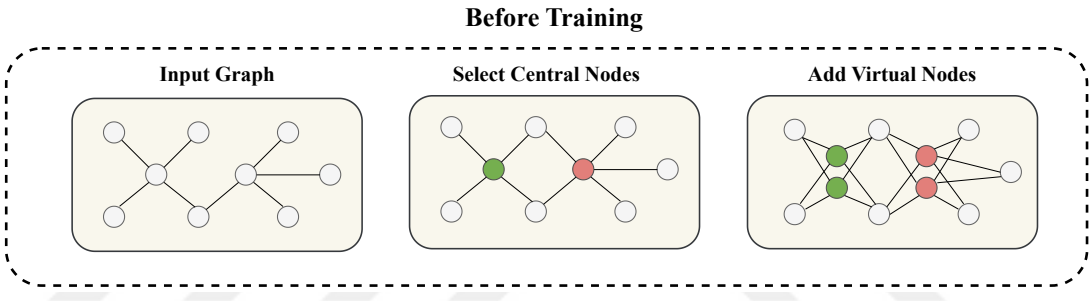
$$\begin{aligned} \mathrm{VN}(G; k) = G_{\mathrm{VN}_k} = (\mathcal{V} \cup \{v_{N+m} \mid m \in [k]\}, \\ \mathcal{E} \cup \bigcup_{i=1}^{k} \{(v_j, v_{N+i}) \mid j \in [N]\}) \end{aligned} \tag{3.28}$$

where k is the number of global virtual nodes.

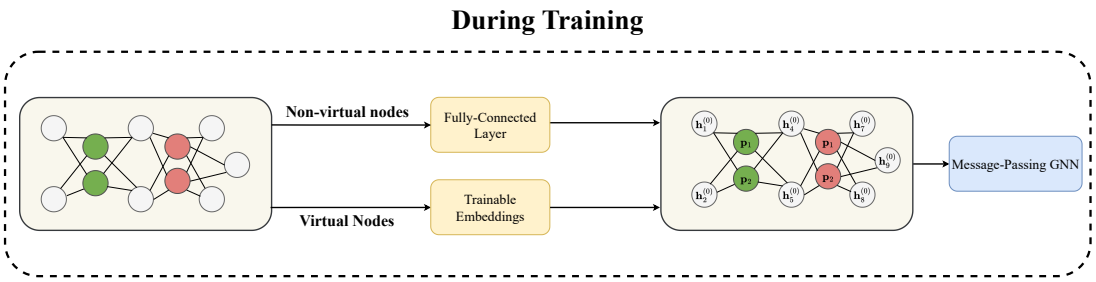
Despite the efficacy of global virtual nodes in creating short circuits for nodes receive long-range information while maintaining linear complexity [63, 66], several

limitations persist that this thesis addresses:

- Global virtual nodes introduce significant computational costs when applied to large graphs by adding additional $k \cdot |V|$ edges to the graph.
- Global virtual nodes inherently modify the global topology, causing it to deviate from the original graph structure, albeit less substantially than graph rewiring techniques.
- Notably large width may be required to store global information [63].



(a) Adding virtual nodes.



(b) Forward pass

Figure 3.2. Procedure for adding virtual nodes to a graph. In (a), we select n_s central nodes. Then, we add n_c virtual nodes for each central while also copying each edge of the central node n_e times. (b) shows the process of assigning trainable embeddings to each virtual node before feeding the graph into the GNN.

3.4.4. Local Virtual Nodes

We propose a local version of virtual nodes as an efficient alternative to graph rewiring and global virtual nodes. As depicted in Figure 3.2, our method contains two procedures: one applied before training and another executed during training. Prior to training, we alter the graph to include virtual nodes to introduce more pathways through central nodes and expand the feature capacity. During training, we equip each virtual node with a trainable embedding to distinguish each virtual node in the same group and enable long-range communication beyond the number of message-passing layers through shared embeddings. Next, we elucidate these two procedures in detail.

3.4.4.1. Virtual Node Integration. In contrast to the literature on virtual node as an entirely new node [63–67], we characterize virtual nodes as support nodes that introduce new pathways through central regions and increase feature capacity of existing nodes.

Initially, we establish a subset of nodes representing common stations connecting most of the nodes. Following Choi *et al.*, we leverage centrality metrics to identify crucial nodes that could benefit from expansion [69]. As elaborated in Section 3.4.5, our approach differs from PANDA in that we exclusively utilize central metrics with linear complexity in terms of number of nodes and edges. We utilize a centrality function, $C : \mathcal{V} \to \mathbb{R}$, to obtain the centrality scores and retrieve n_s nodes with the largest centrality score. We denote this set of nodes as $\mathcal{C}$.

We incorporate three different centrality function based on the following: Degree, Pagerank [77], and Label Propagation [78, 79]. Degree centrality and PageRank centrality functions are implemented as in PANDA [69]. Alongside these standard centrality measures, we implemented a centrality function that applies label propagation to cluster the graph into communities and calculates the out-community degree of each node. Then, for each community, we select the node with highest out-community degree as the central node. The rationale for this criterion is that a node with the maximum number of connections to external communities is more likely to be a source

of a bottleneck.

After determining the set of central nodes, $\mathcal{C}$, we augment each of them with a set of virtual nodes. Specifically, we assign each central node a group of virtual nodes replacing the original node. This group of virtual nodes represents the original node. Number of virtual nodes per central node n_c is a hyperparameter. This hyperparameter may be tuned depending on the topology of the graph in the dataset. The nodes that are in a severe bottleneck region may benefit from a higher number of virtual nodes, while only two virtual nodes may be sufficient for central nodes that are in a well-connected community. Each virtual node in the same group operates independently of each other and receiving information from neighbors of the original node while also propagating features.

Another design choice is how to distribute the edges of the central nodes to their corresponding virtual node group. One option is assigning each edge to a single virtual node. This could be a reasonable approach in scenarios with virtual node groups of small size. However, in the case of a large virtual node groups, each virtual node would have to communicate with other virtual nodes in the same group to receive the features of all neighbors. We may distribute all edges to each virtual node by copying each edge n_c times, allowing a single-hop access to all neighbors. However, this may significantly increase the number of edges if we decide to have many virtual nodes belong to the same group, creating the potential of over-smoothing [52]. Therefore, we introduce an additional hyperparameter, which we call *replication factor* and denote as $n_s \leq n_c$, that indicates the number of edge copies of a central node. The replication factor introduces additional flexibility for our local virtual node frameworks and ensures n_c copies of each edge are stored in the virtual node group. Note that one corner case is when a pair of selected central nodes are adjacent in the original graph. In that case, we simply connect each pair of nodes from the two virtual node groups, creating a complete subgraph. Figure 3.2(a) illustrates the procedure for integrating local virtual nodes into a graph with an example. In practice, we have achieved a substantial performance improvement over graph rewiring and PANDA with little effort put into

the hyperparameter optimization (of n_s, n_c, and n_e), as we discuss in the experiments chapter.

Analogous to global virtual nodes, we can formalize our local virtual nodes framework. Assume that $c(\cdot)$ assigns an index to each central node in $\mathcal{C}$ from the set $[n_s]$. Also, for simplicity, assume that $n_c = n_e$, meaning that each virtual node representing the central node receives all the original edges of the central node. Local virtual node addition with the given hyperparameters can be denoted as

$$\text{LVN}(G; \mathcal{C}, n_c) = G_{\text{LVN}} = (\mathcal{V}_{\text{LVN}}, \mathcal{E}_{\text{LVN}}) \tag{3.29}$$

where $\mathcal{V}_{\text{LVN}}$ and $\mathcal{E}_{\text{LVN}}$ indicate the new set of nodes and edges, respectively. $\mathcal{V}_{\text{LVN}}$ is defined as

$$\mathcal{V}_{\text{LVN}} = \mathcal{V} \cup \bigcup_{j \in \mathcal{C}} \{v_{N+c(j)m} \mid m \in [n_c]\} \tag{3.30}$$

where $\mathcal{V}$ is augmented with $n_s \times n_c$ nodes. Moreover, the updated edge set $\mathcal{E}_{\text{LVN}}$ can be defined as

$$\begin{aligned} \mathcal{E}_{\text{LVN}} \;=\; & \mathcal{E} \cup \bigcup_{\substack{i \in \mathcal{C}, \\ m \in [n_c]}} \{(j, v_{N+c(i)m}) \mid j \in \mathcal{N}(i)\} \\ & \cup \bigcup_{\substack{i,j \in \mathcal{C}, \\ (i,j) \in \mathcal{E}}} \{(v_{N+c(i)y}, v_{N+c(j)z}) \mid y, z \in [n_c]\} \end{aligned} \tag{3.31}$$

where we merge the existing edge set with the virtual node edges originating from central nodes. The first union operator corresponds to assigning the original neighborhood as virtual node neighbors. The second union operator represents edges that emerge from the corner case where the selected central codes are adjacent.

Finally, as we introduce the virtual node groups as a replacement for the central nodes, we remove them from the graph by taking the induced subgraph of G_{LVN} for the set $\mathcal{V} \setminus \mathcal{C}$. More formally, we feed the graph, $G_{\text{LVN}}[\mathcal{V} \setminus \mathcal{C}]$, into the GNN.

In conclusion, local virtual nodes modify the input graph locally to create more pathways through central nodes while introducing virtual nodes that can store additional information in their representations.

3.4.4.2. Trainable Virtual Node Embeddings. Another crucial choice in local virtual node integration is the initial features of virtual nodes. Ideally, we want each virtual node within the same virtual node group to have distinct features, thereby enabling diversity and abundance in central node representations, which would not be possible in a single node case. If we use the original central node features for all nodes within the virtual node group, each virtual node would propagate the same features during message-passing (assuming $n_c = n_e$ scenario) and, therefore, attain the same features. Consequently, we suggest incorporating learnable feature vectors for each virtual node.

In our framework, we assign each virtual node group a set of trainable embeddings, $\mathbf{P}^{n_c \times D}$, where D represents the hidden layer dimensionality of the network. The embeddings $\mathbf{P}$ are designed to be shared across each virtual node group, thereby reducing the computational cost and enforcing generalization. Moreover, the most important advantage of shared trainable embeddings is that they include information from different parts of the graph, which could not be communicated without effective long-range propagation. During training, trainable embeddings are updated using multiple feedback received from possibly non-contiguous regions of the graph. Therefore, the trainable embeddings in a virtual node group contain long-range information independent of the number of message-passing layers. Thus, shared trainable virtual node embeddings allow us to extend the receptive field of GNNs.

Let us formally define how we construct the initial set of node features, $\mathbf{h}^{(0)}$ to be fed into the GNN. We feed non-virtual nodes to a fully-connected layer and assign virtual nodes one of the trainable embeddings. We assign each virtual node in a virtual node group an index from $[n_c]$, which we refer to as *local index*. The local index defines the position of a virtual node within its virtual node group. Assume that the function $\psi : \mathcal{V}_{\text{LVN}} \to \mathbb{Z}^+$ outputs the local index given a virtual node and outputs zero if it is a

non-virtual node. Thus, the initial representation of a node is determined as

$$\mathbf{h}_v^{(0)} = \begin{cases} \mathbf{W}_{\text{in}}^T \mathbf{x}_v + \mathbf{b}_{\text{in}}, & \text{if } \psi(v) = 0 \\ \mathbf{p}_{\psi(v)}, & \text{otherwise} \end{cases} \tag{3.32}$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{F \times D}$, $\mathbf{b}_{\text{in}} \in \mathbb{R}^D$ are the weights and biases of the fully-connected layer, and $\mathbf{p}_{\psi(v)}$ denotes the row of the embedding matrix $\mathbf{P}$ at index $\psi(v)$. After obtaining the initial features $\mathbf{h}_v^{(0)}$, we feed them into a standard message-passing GNN following the framework described in Section 3.2.

3.4.5. Comparison with PANDA

Besides virtual nodes, expanding the width of some selected nodes to mitigate over-squashing has also been explored [69]. In PANDA, the feature dimensionality of a subset of nodes is increased without changing the graph topology. More precisely, k central nodes are determined based on a centrality measure. Selected nodes attain a higher number of features, or width, than the standard width p. Moreover, different message-passing functions are applied for each pair of node types, meaning low width and high width [69]. Our framework has several advantages over PANDA. PANDA leverages the following computationally intensive centrality measures: Betweenness, Closeness, and Load [69]. These centrality measures rely on finding the shortest path between nodes and run in quadratic time [80, 81]. In contrast, we achieve substantial improvement over PANDA by leveraging only the computationally efficient centrality measure that is applicable to large graphs. Further, PANDA necessarily applies different message-passing between nodes of different width [69]. When passing messages from low width to high width, a weight matrix increasing the feature dimensionality is applied. For messages that are passed from high-width to low-width nodes, instead of a fully-connected layer that simply reduces the dimensionality, they incorporate a module that selects k features of high-width nodes. Conversely, our local virtual nodes framework can be combined with any existing GNN in the literature without any modification in the message-passing phase.

4. EXPERIMENTS

This section includes experiments related to CHAT-GNN and local virtual nodes. We report the node classification results of CHAT-GNN on 14 benchmark datasets. The performance of CHAT-GNN is compared with various well-known baselines, including recent studies tackling the over-smoothing problem. In addition, we report the graph classification results of local virtual nodes and various graph rewiring benchmarks.

4.1. Datasets

We utilize node classification datasets for CHAT-GNN, and graph classification datasets for local virtual nodes.

4.1.1. Node Classification

We train CHAT-GNN with commonly used benchmark datasets for the node classification task. Among the benchmark datasets, three citation network datasets with high homophily, Cora, Citeseer, and Pubmed [82] are considered. We use the full split versions of these datasets in which the nodes that are not in validation and test sets are used for training [83]. Chameleon, Squirrel, and Actor, heterophilous Wikipedia networks, are also included [84]. We also consider Cornell, Texas, and Wisconsin, small-scale web networks with high heterophily [84]. Finally, we use a recent set of extremely heterophilous benchmark datasets, namely roman-empire, amazon-ratings, minesweeper, tolokers, and questions designed to evaluate graph representation learning in heterophilous settings [45]. The dataset statistics are provided in Table 4.1 with the homophily metric proposed by Lim *et al.* [85].

We convert all graphs to undirected graphs by adding edges in both directions. Feature normalization that helps stabilize the training process is applied for datasets with bag-of-words features [82, 84]. The node features provided in the datasets by

Platonov *et al.* [45] are used without any modification.

Table 4.1. Statistics of the benchmark datasets including the homophily metric proposed by Lim et al. [85].

dataset	**# nodes**	**# edges**	**# features**	**homophily**
roman-empire	22,662	32,927	300	0.02
amazon-ratings	24,492	93,050	300	0.13
minesweeper	10,000	39,402	7	0.01
tolokers	11,758	519,000	10	0.17
questions	48,921	153,540	301	0.09
chameleon	2,277	36,051	2,325	0.06
squirrel	5,201	216,933	2,089	0.04
cornell	183	295	1,703	0.13
actor	7,600	29,926	932	0.00
texas	183	309	1,703	0.00
wisconsin	251	499	1,703	0.08
cora	2,708	10,556	1,433	0.77
citeseer	3,327	9,104	3,703	0.63
pubmed	19,717	88,648	500	0.66

4.1.2. Graph Classification

We train GCNs combined with our local virtual nodes framework on the graph classification task. We use the following graph classification datasets from the TU-Dataset [86]: REDDIT-BINARY, IMDB-BINARY, MUTAG, PROTEINS, ENZYMES, and COLLAB. REDDIT-BINARY, IMDB-BINARY, and COLLAB contain social networks. ENZYMES and PROTEIONS are bioinformatics datasets comprised of macromolecules represented as graphs [86]. MUTAG is a dataset of small molecules [86]. For the REDDIT-BINARY, IMDB-BINARY, and COLLAB datasets, which do not contain

any node features, we add a single constant feature with a value of one, following the experimental setup of FoSR and PANDA [55, 69]. The statistics of the datasets are provided in Table 4.2.

Table 4.2. Statistics of the graph classification benchmark datasets.

dataset	# graphs	avg. degree	avg. # nodes	avg. # edges	avg. diameter
REDDIT-BINARY	2000	2.34	429.63	995.51	8.59
IMDB-BINARY	1000	8.89	19.77	193.06	1.86
MUTAG	188	2.19	17.93	39.59	8.22
ENZYMES	600	3.86	32.63	124.27	10.92
PROTEINS	1113	3.73	39.06	145.63	11.62
COLLAB	5000	37.37	74.49	4914.43	1.86

4.2. Baselines

We have a diverse set of baselines from the GNN literature to compare with CHAT-GNN. GCN [10], GAT [11], and GATv2 [17] are included as the standard models in the literature. GCN-II [26] and Jumping Knowledge Network (JKNet) [87] are early attempts at deep learning on graphs. H2GCN [44] is an improvement in learning on graphs with high heterophily. FAGCN [31] tries to overcome over-smoothing with high-pass filters. AERO-GNN [27] and Ordered GNN [43] are recent models that can scale to many layers and mitigate over-smoothing. We also include a Multilayer Perceptron (MLP), which only uses the node features, to have an idea of how much it can learn without using the graph structure.

For local virtual nodes, we report graph classification performance of well-known graph rewiring methods, DIGL [50], SDRF [51], FoSR [55] and BORF [52], and the width expansion method PANDA [69]. Additionally, as suggested by Alon and Yahav [24], we also implement a GCN that treats the graph as a complete graph at the last layer.

4.3. Implementation Details

In this section, we provide the implementation details of node classification and graph classification experiments.

4.3.1. Node Classification

We train all the models from scratch with Adam optimization algorithm [88]. The maximum number of epochs is set to 5000 with the early stopping patience of 500 epochs starting after 200 warm-up epochs. The early stopping is determined based on the validation accuracy. Weight decay is used to regularize the training. We tune the learning rate, weight decay rate, hidden dimension, number of layers, and model-specific hyperparameters using Bayesian optimization. We run 75 iterations of Bayesian optimization for each model and dataset. We evaluate the models using the average validation accuracy of the first three splits. Finally, we select the hyperparameter settings with the highest validation accuracy and re-train the models with the same configuration using the ten dataset splits.

The implementation of the proposed CHAT-GNN and baselines used in the experiments of this study with the best hyperparameters and their search spaces is publicly available at [1] . The official implementations of the recent baseline models, AERO-GNN [2] and Ordered GNN [3] are used. We run both models with the reported best hyperparameters instructed by the authors for the common benchmark datasets. We use the implementations in the popular graph learning library called PyTorch Geometric [89] for other baseline models. When the performance on some of the benchmark datasets is missing in the baseline studies, we train the baseline models from scratch by running hyperparameter optimization considering the suggested hyperparameter setting by the authors, dataset scale, and our computational resources. We train all the models on a single GPU. We use NVIDIA Tesla T4 (16GiB) and NVIDIA Tesla V100 (16GiB) as

[1] `https://github.com/ALLab-Boun/CHAT-GNN`
[2] `https://github.com/syleeheal/AERO-GNN`
[3] `https://github.com/LUMIA-Group/OrderedGNN`

our GPU hardware.

4.3.2. Graph Classification

Similarly to node classification, we train the models from scratch using the Adam optimization algorithm. The models are trained for a maximum of 300 epochs with early stopping if no improvement occurs within 100 epochs. To ensure comparability, we use the same setup described by Karhadkar *et al.* [55], which was also adopted by Choi *et al.* [69]. Specifically, we set the learning rate to 10^{-3}, weight decay rate to 10^{-5}, number of hidden GNN layers to 4, and hidden dimension to 64. For DIGL and SDRF, we use the best hyperparameters reported by Karhadkar *et al.* [55]. For FoSR[4] , BORF[5] , and PANDA[6] , we use the official implementations and hyperparameters provided by the authors. Although original implementations directly feed the input features to GNN, we include an initial fully-connected layer for fair comparison with our trainable virtual node embeddings. We train all the models on a single NVIDIA A40 (48GiB) GPU.

4.4. CHAT-GNN Results

4.4.1. Overall Performance

Node classification performances of all baselines and CHAT-GNN are shown in Tables 4.3, 4.4 and 4.5. The results in this table are obtained by training all the baseline models from scratch. The datasets minesweeper, tolokers, and questions by Platonov *et al.* [45] constitute binary classification problems. As the authors suggested, we report AUC for minesweeper, tolokers, and questions. For other datasets, we report mean classification accuracy. All datasets have ten train, validation, and test splits. We report the average performance with its standard deviation.

[4] `https://github.com/kedar2/FoSR`
[5] `https://github.com/hieubkvn123/revisiting-gnn-curvature`
[6] `https://github.com/jeongwhanchoi/PANDA`

Table 4.3. Node classification performance on highly heterophilous graphs introducted by Platonov et al. [45]

Model	**roman-emp.**	**amazon-rat.**	**minesw.**	**tolokers**	**questions**
MLP	67.0 ± 0.6	45.2 ± 0.5	51.0 ± 1.3	71.9 ± 0.7	70.5 ± 1.0
GCN	51.9 ± 0.5	48.4 ± 0.5	72.3 ± 1.0	75.2 ± 1.3	76.3 ± 1.0
GAT	60.0 ± 0.9	45.9 ± 0.7	86.3 ± 0.9	74.4 ± 0.8	70.9 ± 1.2
GATv2	52.1 ± 1.9	44.2 ± 0.6	73.4 ± 1.0	73.4 ± 0.9	73.3 ± 1.0
GCN-II	82.5 ± 0.7	52.2 ± 0.5	90.5 ± 0.6	77.5 ± 1.0	74.6 ± 0.9
JKNet	79.2 ± 0.6	46.5 ± 0.5	88.2 ± 0.7	80.7 ± 1.3	77.5 ± 0.9
H2GCN	79.5 ± 0.6	50.2 ± 0.4	90.2 ± 0.5	OOM	71.3 ± 1.1
FAGCN	71.5 ± 0.5	48.4 ± 0.7	87.8 ± 0.9	77.4 ± 1.2	75.9 ± 1.6
AERO-GNN	80.5 ± 0.9	51.5 ± 0.7	$\underline{93.0 \pm 0.9}$	81.8 ± 0.9	77.0 ± 1.1
Ordered GNN	$\underline{85.1 \pm 0.4}$	$\mathbf{53.6 \pm 0.5}$	92.0 ± 0.6	$\underline{84.6 \pm 0.5}$	$\mathbf{77.9 \pm 1.2}$
CHAT-GNN	$\mathbf{91.3 \pm 0.4}$	$\underline{52.6 \pm 0.6}$	$\mathbf{97.3 \pm 0.3}$	$\mathbf{85.7 \pm 0.7}$	$\underline{77.8 \pm 1.2}$

Table 4.4. Node classification performance on heterophilous Wikipedia and web networks [84].

model	**chameleon**	**squirrel**	**cornell**	**actor**	**texas**	**wisconsin**
MLP	49.6 ± 2.3	33.2 ± 1.1	73.3 ± 7.1	35.6 ± 1.5	78.9 ± 3.8	85.3 ± 3.2
GCN	64.7 ± 2.1	47.7 ± 1.9	50.0 ± 9.2	30.6 ± 0.5	63.8 ± 4.2	62.2 ± 5.1
GAT	68.1 ± 1.7	52.0 ± 1.4	45.9 ± 7.1	29.8 ± 1.0	59.7 ± 6.4	64.9 ± 5.2
GATv2	63.9 ± 1.5	47.2 ± 1.5	46.2 ± 7.3	29.2 ± 1.0	61.4 ± 5.9	61.2 ± 4.3
GCN-II	66.1 ± 2.0	49.4 ± 1.4	68.7 ± 7.8	$\underline{36.0 \pm 0.9}$	76.8 ± 6.9	83.5 ± 5.2
JKNet	66.1 ± 1.6	54.0 ± 1.5	73.5 ± 3.1	35.3 ± 0.6	79.2 ± 5.0	83.1 ± 4.5
H2GCN	58.0 ± 1.6	33.4 ± 1.9	71.9 ± 5.6	35.5 ± 1.1	82.2 ± 5.2	$\underline{86.3 \pm 3.6}$
FAGCN	61.0 ± 1.5	38.6 ± 2.5	$\underline{76.2 \pm 2.6}$	35.3 ± 0.9	83.3 ± 6.4	81.4 ± 4.2
AERO-GNN	$\mathbf{71.7 \pm 1.9}$	$\underline{61.9 \pm 2.3}$	$\mathbf{76.6 \pm 4.5}$	$\mathbf{36.5 \pm 1.3}$	$\underline{83.8 \pm 4.7}$	83.7 ± 3.7
Ordered GNN	$\underline{71.2 \pm 1.5}$	60.1 ± 2.3	74.9 ± 4.0	35.1 ± 1.3	80.3 ± 4.8	86.1 ± 3.5
CHAT-GNN	70.5 ± 2.1	$\mathbf{64.3 \pm 1.5}$	72.2 ± 4.2	35.6 ± 1.3	$\mathbf{84.3 \pm 5.4}$	$\mathbf{87.3 \pm 3.2}$

Table 4.5. Node classification performance on benchmark citation network datasets with high homophily [82].

Model	**cora**	**citeseer**	**pubmed**
MLP	76.9 ± 0.0	75.4 ± 0.0	89.1 ± 0.0
GCN	87.3 ± 0.3	78.7 ± 0.4	88.7 ± 0.2
GAT	$\underline{87.6 \pm 0.4}$	78.6 ± 0.4	87.0 ± 0.3
GATv2	$\underline{87.6 \pm 0.4}$	78.5 ± 0.3	86.7 ± 0.3
GCN-II	86.7 ± 0.5	78.4 ± 0.5	$\mathbf{90.6 \pm 0.5}$
JKNet	85.9 ± 0.5	77.8 ± 0.7	89.4 ± 0.3
H2GCN	87.2 ± 0.3	78.0 ± 0.4	90.1 ± 0.1
FAGCN	$\mathbf{88.4 \pm 0.3}$	78.3 ± 0.5	90.3 ± 0.2
AERO-GNN	87.4 ± 0.7	78.7 ± 0.7	$\underline{90.5 \pm 0.4}$
Ordered GNN	86.4 ± 0.5	$\underline{78.8 \pm 1.0}$	$\mathbf{90.6 \pm 0.3}$
CHAT-GNN	87.0 ± 0.5	$\mathbf{79.0 \pm 0.4}$	$\mathbf{90.6 \pm 0.3}$

In addition, we compare the performance of CHAT-GNN with the best performance directly reported by the AERO-GNN and Ordered GNN papers in Table 4.6. As seen in Tables 4.3, 4.4 and 4.5 and 4.6, CHAT-GNN outperforms baseline methods on heterophilous graphs such as minesweeper, roman-empire, and squirrel. For citation networks with high homophily, CHAT-GNN is either comparable to the state-of-the-art or outperforms by a small margin. We can observe that the proposed model adapts to graphs of different characteristics in terms of domain, homophily, and size by achieving either superior or comparable performance with the well-known GNNs in the literature.

Table 4.6. Comparison with the best performances reported by AERO-GNN [27] and Ordered GNN [43] studies.

model	**AERO-GNN**	**Ordered GNN**	**CHAT-GNN**
chameleon	$\underline{71.6 \pm 2.4}$	$\mathbf{72.3 \pm 2.3}$	70.5 ± 2.1
squirrel	61.8 ± 2.4	$\underline{62.4 \pm 2.0}$	$\mathbf{64.3 \pm 1.5}$
cornell	$\underline{81.2 \pm 6.8}$	$\mathbf{87.0 \pm 4.7}$	72.2 ± 4.2
actor	$\underline{36.6 \pm 1.1}$	$\mathbf{38.0 \pm 1.0}$	35.6 ± 1.3
texas	$\underline{84.3 \pm 5.2}$	$\mathbf{86.2 \pm 4.1}$	$\underline{84.3 \pm 5.4}$
wisconsin	84.8 ± 3.3	$\mathbf{88.0 \pm 3.7}$	$\underline{87.3 \pm 3.2}$

Table 4.7. Best performing layer of our model on each benchmark dataset.

dataset	**best layer**	**search space**
roman-empire	10	{2, 4, 6, 8, 10, 12}
amazon-ratings	6	{2, 3, 4, 5, 6}
minesweeper	14	{2, 4, 6, 8, 10, 12, 14}
tolokers	10	{2, 3, 4, 5, 6, 7, 8, 9, 10}
questions	10	{2, 4, 6, 8, 10, 12}
chameleon	12	{2, 4, 6, 8, 10, 12, 14, 16, 18}
squirrel	12	{2, 4, 6, 8, 10, 11, 12, 13, 14}
cornell	2	{2, 4, 6, 8, 10}
actor	8	{2, 4, 6, 8}
wisconsin	2	{2, 3, 4, 5}
cora	5	{2, 3, 4, 5}
citeseer	4	{2, 3, 4, 5}
pubmed	2	{2, 3, 4, 5}

4.4.2. Number of Layers vs. Performance

A significant decrease in performance after a certain number of message-passing layers is a clear indication of over-smoothing in GNNs [20, 21]. To investigate the over-smoothing in the proposed framework, we train CHAT-GNN and several message-passing GNNs with 2 to 32 layers on three datasets with differing characteristics. Figure 4.1 shows the performance change with the increasing number of layers. We also use 32 as the number of hidden channels for all models without additional tuning. The results show that while standard models such as GCN and GAT suffer from over-smoothing at the early layers, the proposed model is able to mitigate over-smoothing and compete with AERO-GNN that combines edge attention and hop attention [27].

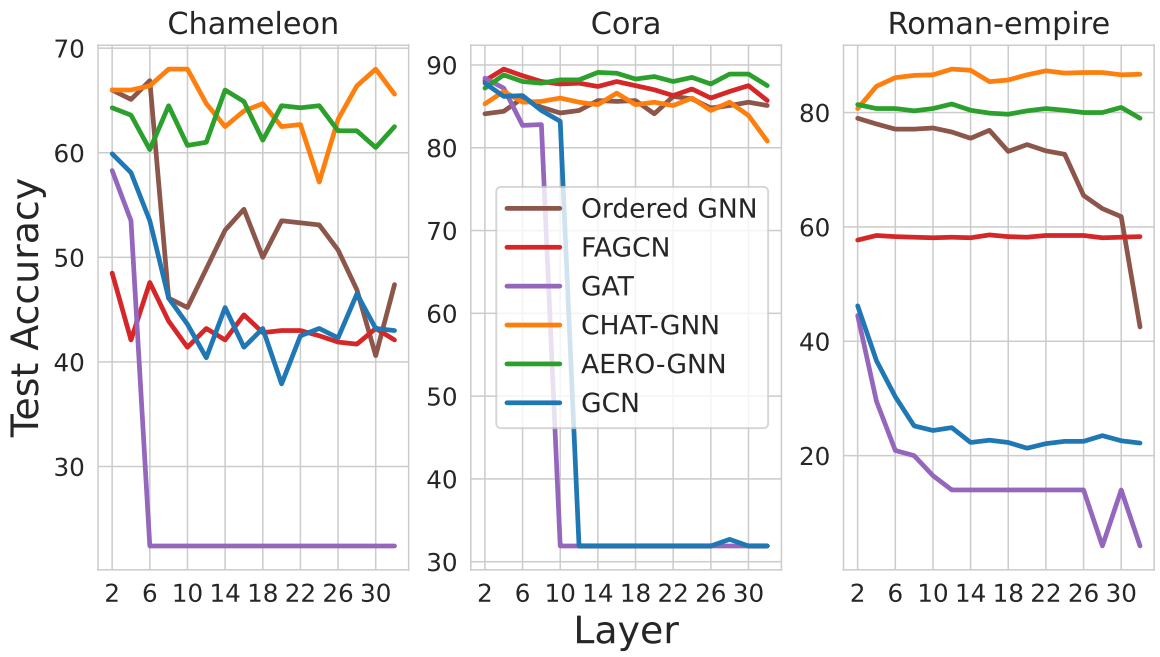


Figure 4.1. Resistance of selected models to over-smoothing. The change in test accuracy is plotted with respect to an increasing number of message-passing layers.

We also report the best-performing number of layers of CHAT-GNN for each dataset, along with the search space for the number of layers in Table 4.7. The table shows that CHAT-GNN achieves its best performance on layers that classical GNNs

suffer from over-smoothing. The best performance is observed on graphs with high homophily with fewer layers. This shows the capability of CHAT-GNN to adjust itself to the high homophily setting while achieving state-of-the-art performance on graphs with high heterophily. We select different search spaces for each dataset by considering dataset characteristics and hardware limitations.

4.4.3. Dirichlet Energy

Dirichlet energy, defined in Equation (3.17) is used to measure how much a function changes. In our case, it can be used to measure how the features of nodes in a graph change. In GraphCON study [75], the authors use the Dirichlet energy to quantify over-smoothing as the exponential decay of the Dirichlet energy of the hidden node features as we go deeper through the message-passing layers in a GNN [75].

We use the same setting in GraphCON [75] to measure the Dirichlet energy of the output features of message-passing layers of various GNNs. Namely, we simulate message-passing on a 10×10 grid, in which each node has four neighbors. Each node has two features sampled from the uniform distribution, $\mathcal{U}(0, 1)$. We run message-passing on layers with random weights up to 1000 layers. Figure 4.2 shows how the Dirichlet energy changes with respect to the layers. The proposed CHAT-GNN keeps the Dirichlet energy almost constant and higher than AERO-GNN, while the baseline models reach almost zero Dirichlet energy in deeper layers, which indicates severe over-smoothing [75]. The figure shows the scalability of CHAT-GNN to very deep message-passing layers.

4.4.4. Visualizing Channel Weights

We visualize the weight vectors β_{ji} through message-passing layers in CHAT-GNN in Figure 4.3. As discussed in Subsection 3.3.1, β_{ji} denotes the channel-wise attention vector to evaluate the message function from node i to node j. We visualize the pairwise cosine similarities, $\frac{\beta_{ji}^T \beta_{ki}}{\|\beta_{ji}\| \|\beta_{ki}\|}$, between each neighbor i in Figure 4.3. If the

cosine similarity is close to one for an edge pair (j, i) and (k, i), the nodes k and j receive almost the same features from the node i. The results show that CHAT-GNN can learn to apply different message-passing schemes for different neighbors, nodes, and hops. Furthermore, while it sends diverse messages to neighbors in heterophilous graphs, it sends similar messages to neighbors in graphs with high homophily. However, if needed, it adjusts the message-passing mechanism to send more diverse messages. Note that we randomly draw five nodes from each dataset for illustration purposes. Also, we visualize the first five layers. Refer to Appendix A for illustrations at deeper layers and more datasets.

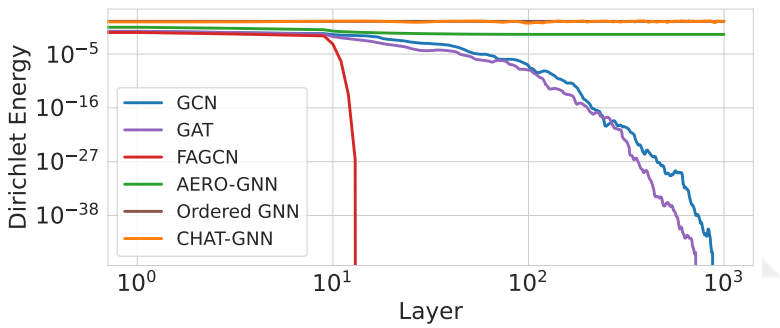


Figure 4.2. Change in Dirichlet energy of the message-passing layer output features of selected models with respect to increasing number of layers.

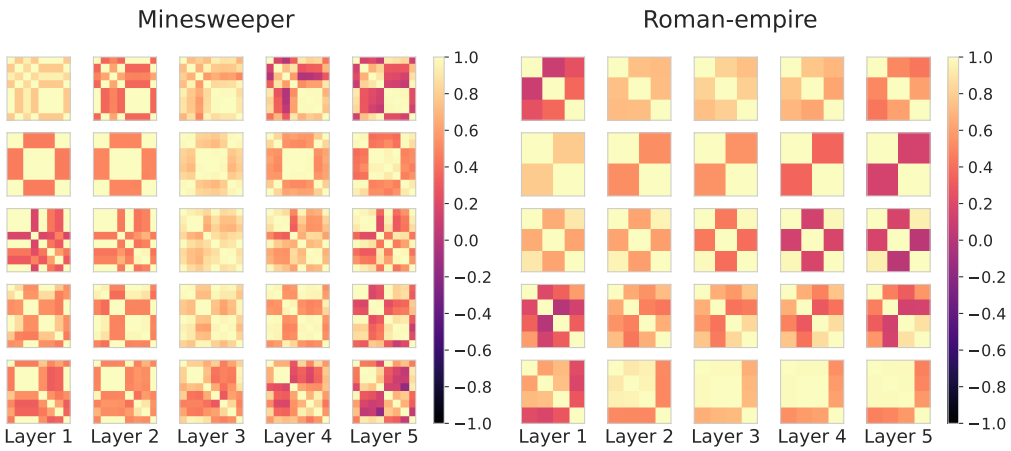


Figure 4.3. Heatmap of pairwise cosine similarities between β_{ji}'s for a randomly selected node i and its neighbor j throughout the message-passing layers of CHAT-GNN.

4.4.5. Directed GNN Setting

Edge directions are often not considered in the baseline GNNs. A message-passing layer is concerned with only the immediate neighbors of a node. It becomes a common practice to convert the graph to an undirected one to include both directions in message-passing. However, recent studies, such as Dir-GCN, suggest applying message-passing in both directions separately and with separate learnable parameters [90]. The bidirectional message-passing results show that when combined with Jumping Knowledge Networks [87] and a GCN or GAT layer as the base message-passing layer, the performance is boosted on some heterophilous graphs [90]. Therefore, we also report the bidirectional message-passing results of CHAT-GNN in Table 4.8 on three heterophilous graphs. The results are obtained after hyperparameter tuning on Dir-GCN and Dir-CHAT-GNN. Table 4.8 shows that adding edge directions improves the predictive performance of CHAT-GNN more than the standard GCN. Note that, unlike Dir-GCN, we achieve competitive results without employing Jumping Knowledge Networks in Dir-CHAT-GNN, which adds computational overhead.

Table 4.8. Performance comparison of CHAT-GNN and standard GCN under bidirectional message-passing setting.

model	**roman-empire**	**tolokers**	**chameleon**
Dir-GCN	90.7 ± 0.5	77.7 ± 1.1	$\mathbf{78.5 \pm 1.1}$
Dir-CHAT-GNN	$\mathbf{94.3 \pm 0.4}$	$\mathbf{83.1 \pm 0.9}$	76.6 ± 2.5

4.4.6. Ablation Study

We conduct an ablation study on layer normalization and separate learnable projection modules of CHAT-GNN. Instead of a setting with a fixed number of layers, we train CHAT-GNN models with varying numbers of message-passing layers and visualize the effect of a specific component.

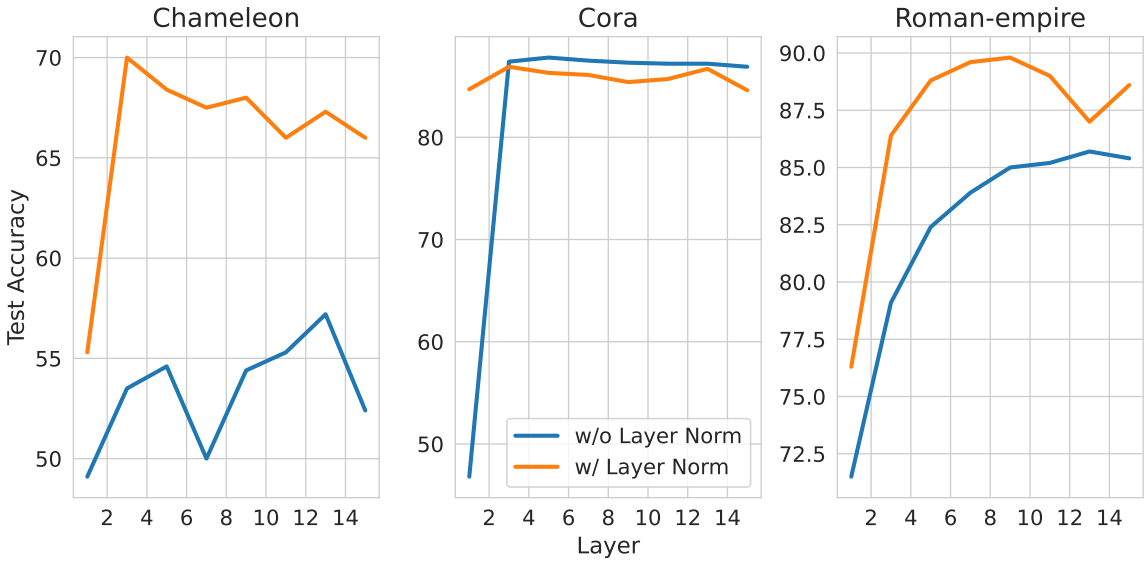


Figure 4.4. Effect of layer norm on the test accuracy as the number of layers increases.

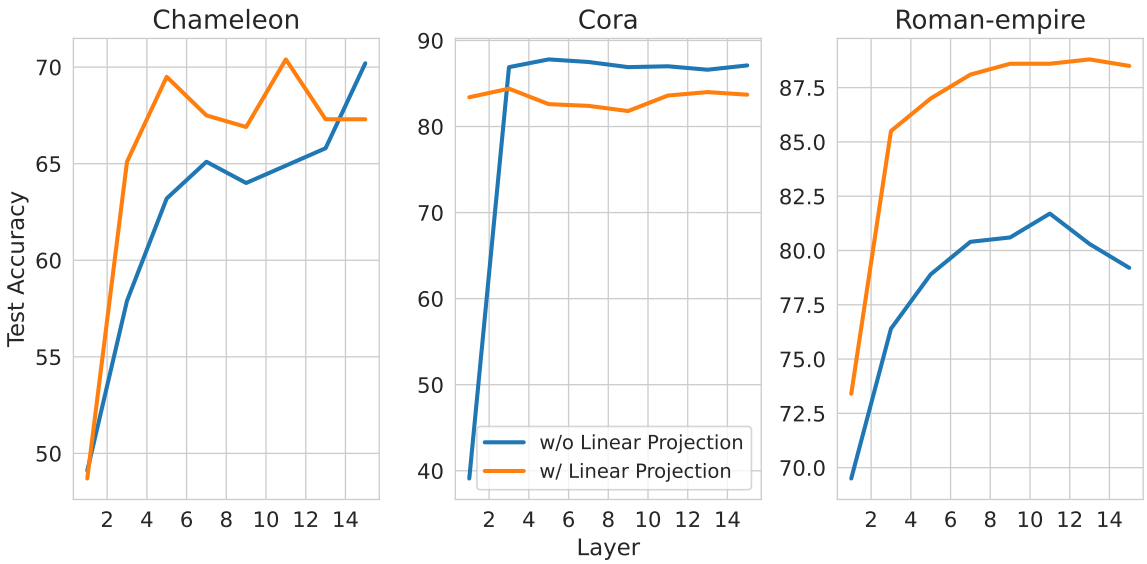


Figure 4.5. Effect of the separate learnable projection layers on the test accuracy as the number of layers increases.

4.4.6.1. Layer Normalization. We observe that layer normalization usually has a positive effect on classification performance, as seen in Figure 4.4. It rescales the combined node and neighborhood representation, and intuitively, this may help the optimization

process. For the citation network dataset Cora, we observe that adding layer normalization slightly reduces the accuracy. This is because the citation network datasets Cora, Citeseer, and Pubmed can easily be overfitted, as observed in all the baselines.

4.4.6.2. Linear Projection. We investigate the effect of adding separate learnable linear projection layers to the node and the neighborhood representations in Figure 4.5. For Chameleon, we observe that the model without projection layers eventually surpasses the models with projection layers. In Cora, we see an effect similar to layer normalization. For the roman-empire, we see a substantial improvement with linear projection layers.

4.5. Local Virtual Nodes Results

4.5.1. Overall Performance

Graph classification performances of baselines that combat over-squashing can be observed in Table 4.9. As there are no official splits, we randomly split each dataset into 80% train, 10% validation, and 80% test sets. All models were trained from scratch, with results reported as mean test accuracy with 95% confidence intervals across 50 runs.

Table 4.9. Performance comparison of baselines and local virtual nodes (LVN) on graph classification datasets. All methods are combined with the same GCN architecture.

Model	REDDIT-BINARY	IMDB-BINARY	MUTAG	ENZYMES	PROTEINS	COLLAB
None	56.530 ± 2.603	50.020 ± 1.309	71.400 ± 3.125	28.033 ± 2.019	71.410 ± 1.077	48.556 ± 2.339
Last Layer FA	58.680 ± 2.421	49.900 ± 1.574	69.400 ± 3.245	27.267 ± 1.810	71.964 ± 1.362	51.140 ± 1.107
DIGL	50.440 ± 0.938	50.280 ± 1.261	76.000 ± 3.072	27.467 ± 1.682	70.929 ± 1.290	32.920 ± 5.231
SDRF	62.720 ± 2.701	49.760 ± 1.338	73.300 ± 2.617	28.533 ± 1.616	71.054 ± 0.925	46.548 ± 2.576
FoSR	70.020 ± 0.911	50.120 ± 1.263	80.400 ± 2.430	23.567 ± 1.313	72.071 ± 1.163	48.772 ± 2.308
BORF	OOM	47.280 ± 1.473	76.500 ± 3.397	25.767 ± 1.711	70.339 ± 0.918	OOM
PANDA	$\underline{80.080 \pm 1.241}$	$\mathbf{64.080 \pm 1.319}$	$\underline{83.000 \pm 1.876}$	$\underline{32.067 \pm 1.856}$	$\mathbf{75.750 \pm 0.944}$	$\underline{68.060 \pm 0.616}$
LVN	$\mathbf{85.230 \pm 0.794}$	$\underline{63.900 \pm 1.530}$	$\mathbf{84.555 \pm 2.450}$	$\mathbf{32.867 \pm 1.568}$	$\underline{74.126 \pm 1.157}$	$\mathbf{72.212 \pm 0.787}$

Table 4.9 shows that our framework outperforms established graph rewiring methods designed to address bottlenecks and the state-of-the-art width-expansion approach PANDA by a large margin.

4.5.2. Effective Resistance

In addition to benchmarking on graph classification, we evaluate our local virtual nodes approach using a structural metric called *effective resistance*. In electrical networks, effective resistance between two points refers to the potential difference if a unit current is applied between them [91]. In graph theory, effective resistance between two nodes quantifies their connectivity [53]. Effective resistance has been leveraged in the graph rewiring literature [52, 54]. G-RLEF, proposed by Banerjee *et al.*, employs an effective resistance-based sampling to identify node pairs for edge addition [54]. Furthermore, Black *et al.* theoretically proved that effective resistance is related to over-squashing and proposed a rewiring algorithm that reduces total effective resistance [53].

Effective resistance between a pair of nodes u and v is formally defined as

$$\mathcal{R}_{u,v} = (\mathbf{1}_u - \mathbf{1}_v)^T \mathbf{L}^+ (\mathbf{1}_u - \mathbf{1}_v) \tag{4.1}$$

where $\mathbf{1}_u$ denotes with entries all zero but one at u and $\mathbf{L}^+$ is the Moore-Penrose pseudoinverse of the Laplacian matrix [53, 54]. Then the total effective resistance of a graph is defined as

$$\mathcal{R}_{\text{tot}}(G) = \frac{1}{2} \sum_{u,v \in \mathcal{V}} \mathcal{R}_{u,v} = N \sum_{i=2}^{N} \frac{1}{\lambda_i} \tag{4.2}$$

where λ_i denotes the i-th eigenvalue of the Laplacian $\mathbf{L}$ [91]. Black *et al.* computes total effective resistance as new edges are added to the graph [53]. Since $\mathcal{R}_{\text{tot}}(G)$ increases with the number of nodes, it would be unfair to compare the total effective resistance as the number of virtual nodes increases. Therefore, we compute a modified version of total effective resistance. To ensure a fair comparison, we determine a subset

of nodes S that includes the non-virtual nodes when the number of selected nodes to add virtual nodes is maximum. Specifically, we independently transform the graph to include virtual nodes for the following set of n_s values: $\{1, 2, 3, 5, 7, 10, 12, 15\}$. Let $\mathcal{C}$ denote the set of 15 most central nodes. We compute the total effective resistance between each pair of nodes in S for each case, where $S = \mathcal{V} \setminus \mathcal{C}$.

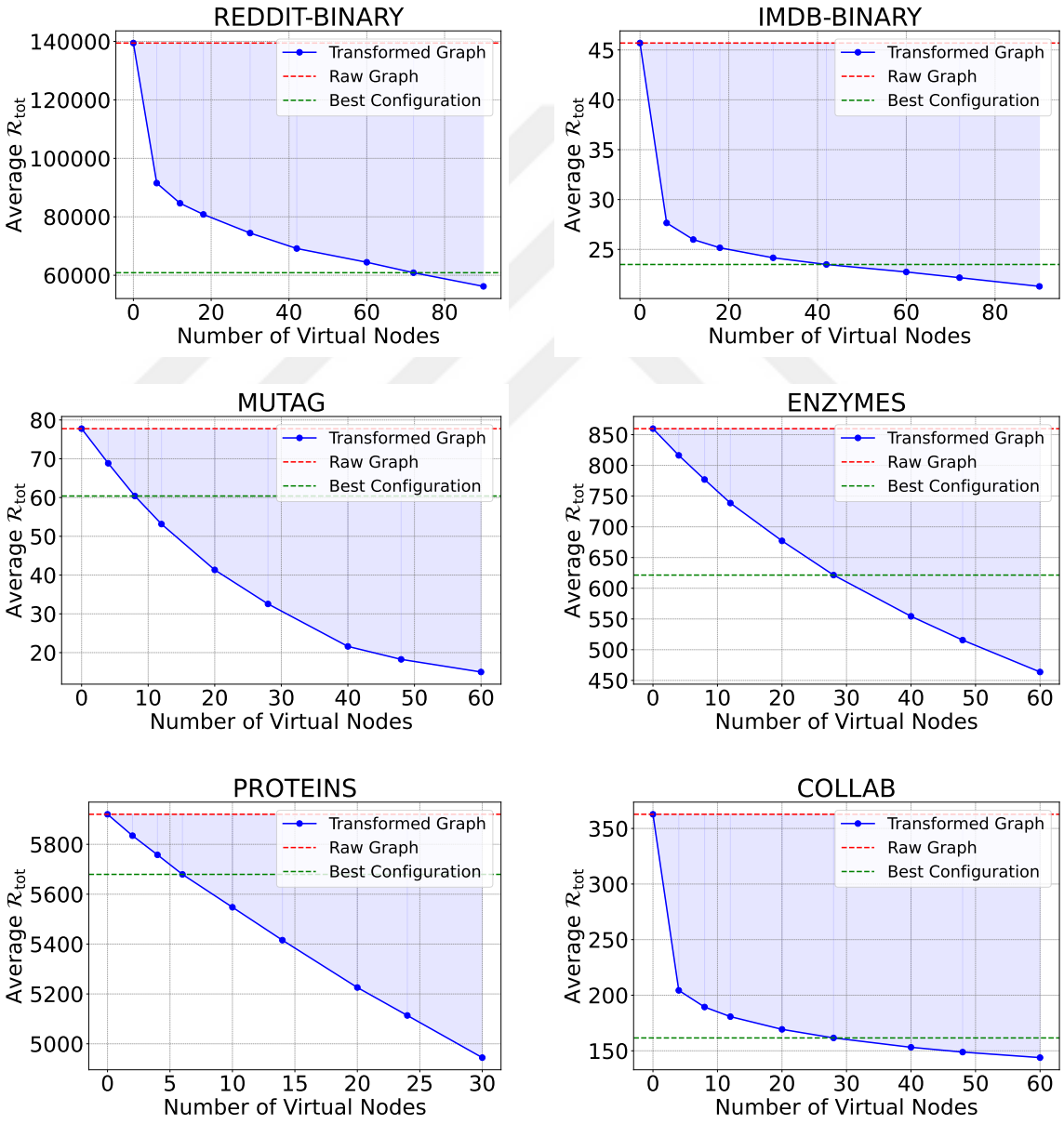


Figure 4.6. Change in the total effective resistance of the non-virtual node set for TUDataset [86].

Figure 4.6 shows our virtual node integration approach improves the connectivity between non-central nodes by reducing the effective resistance. We see a substantial reduction in total effective resistance with only a few virtual nodes for social network datasets since they usually contain a small subset of central nodes connecting several communities. On the other hand, bioinformatics and small molecule datasets have path-like graphs with uniform degree distribution. Therefore, in these datasets, we see a monotonic decrease in total effective resistance. Refer to the Appendix B for example graph visualization examples with and without virtual nodes.

5. CONCLUSION

In this thesis, we address over-smoothing and over-squashing, two critical limitations that impede GNN training. We propose an attention-based GNN architecture called CHAT-GNN that mitigates over-smoothing by attending to each neighbor's feature channels. Experiments show its superiority and ability to alleviate over-smoothing. We also visualize the learned attention vectors that are used in channel-wise message-passing. The illustrations show that the proposed model can adjust the weights according to the node's neighbors and hops. We propose a local version of the virtual nodes idea that has been studied in the literature [14, 16, 64]. Our local virtual nodes framework improves the representation capacity of a central node while also introducing new pathways to the graph to increase its connectivity. Local virtual nodes achieve substantial improvement over the baselines that aim to mitigate over-squashing without semantically distorting the input graph representation. In addition to the task performance, we show that our approach strengthens the connectivity of the graph through effective resistance analysis. Moreover, our shared trainable virtual node embeddings enable an information communication radius far greater than the receptive of the GNN.

In future work, the scalability of the CHAT-GNN and local virtual nodes will be investigated by applying them to large-scale benchmarks such as Open Graph Benchmark [92] and Long Range Graph Benchmark [33]. The proposed channel-wise attention and local virtual node mechanisms could stand out even more in large-scale settings where longer-range relationships must be captured. Additionally, we will evaluate CHAT-GNN on graph-level datasets such as TUDataset [86], while benchmarking the local virtual nodes framework against node classification datasets. To further validate our approach, we will conduct supplementary experiments measuring long-range signal propagation and testing on synthetic tasks like Tree-Neighbors [24, 25]. Finally, we will perform theoretical analyses to demonstrate that our virtual node integration and shared virtual node embedding methods effectively mitigate over-squashing.

REFERENCES

1. Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec and P. Battaglia, "Learning to Simulate Complex Physics with Graph Networks", *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119, pp. 8459–8468, Virtual, 2020.

2. van den Berg, R., T. N. Kipf and M. Welling, "Graph Convolutional Matrix Completion", ArXiv:1706.02263 [stat.ML], 2017.

3. He, X., K. Deng, X. Wang, Y. Li, Y. Zhang and M. Wang, "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation", *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, Virtual Event, China, 2020.

4. Yao, L., C. Mao and Y. Luo, "Graph Convolutional Networks for Text Classification", *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 7370–7377, Honolulu, Hawaii, USA, 2019.

5. Yasunaga, M., H. Ren, A. Bosselut, P. Liang and J. Leskovec, "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering", *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 535–546, Virtual, 2021.

6. Shi, L., Y. Zhang, J. Cheng and H. Lu, "Skeleton-Based Action Recognition With Directed Graph Neural Networks", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7904–7913, Long Beach, California, 2019.

7. Johnson, J., A. Gupta and L. Fei-Fei, "Image Generation from Scene Graphs", *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1219–1228, Salt Lake City, Utah, 2018.

8. Hermosilla, P., M. Schäfer, M. Lang, G. Fackelmann, P. P. Vázquez, B. Kozlíková, M. Krone, T. Ritschel and T. Ropinski, "Intrinsic-Extrinsic Convolution and Pooling for Learning on 3D Protein Structures", *International Conference on Learning Representations*, Virtual, 2021.

9. Xia, T. and W.-S. Ku, "Geometric Graph Representation Learning on Protein Structure Prediction", *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1873–1883, Virtual Event, Singapore, 2021.

10. Kipf, T. N. and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", *Proceedings of the 5th International Conference on Learning Representations*, pp. 2713–2727, Toulon, France, 2017.

11. Veličković, P., G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks", *Proceedings of the 6th International Conference on Learning Representations*, pp. 2920–2931, Vancouver, Canada, 2018.

12. Defferrard, M., X. Bresson and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering", *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 3844–3852, Barcelona, Spain, 2016.

13. Hamilton, W. L., R. Ying and J. Leskovec, "Inductive Representation Learning on Large Graphs", *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, Long Beach, California, USA, 2017.

14. Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, "Neural Message Passing for Quantum Chemistry", *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, pp. 1263–1272, Sydney, NSW, Australia, 2017.

15. MacKay, D. J. C., *Information Theory, Inference, and Learning Algorithms*, Copyright Cambridge University Press, Cambridge, 7.2 edn., 2005.

16. Battaglia, P., J. B. C. Hamrick, V. Bapst, A. Sanchez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. Allen, C. Nash, V. J. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li and R. Pascanu, "Relational Inductive Biases, Deep Learning, and Graph Networks", ArXiv:1806.01261 [cs.LG], 2018.

17. Brody, S., U. Alon and E. Yahav, "How Attentive are Graph Attention Networks?", *International Conference on Learning Representations*, Virtual, 2022.

18. Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, " Going Deeper with Convolutions ", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, Los Alamitos, CA, USA, 2015.

19. He, K., X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, Nevada, 2016.

20. Li, Q., Z. Han and X.-M. Wu, "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning", *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, pp. 3538–3545, New Orleans, Louisiana, USA, 2018.

21. Oono, K. and T. Suzuki, "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification", *Proceedings of the 8th International Conference on Learning Representations*, pp. 12344–12380, Addis Ababa, Ethiopia, 2020.

22. NT, H. and T. Maehara, "Revisiting Graph Neural Networks: All We Have is Low-Pass Filters", ArXiv:1905.09550 [stat.ML], 2019.

23. Chen, D., Y. Lin, W. Li, J. Zhou and X. Sun, "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 3438–3445, New York, USA, 2020.

24. Alon, U. and E. Yahav, "On the Bottleneck of Graph Neural Networks and its Practical Implications", *International Conference on Learning Representations*, Virtual, 2021.

25. Di Giovanni, F., L. Giusti, F. Barbero, G. Luise, P. Liò and M. Bronstein, "On Over-Squashing in Message Passing Neural Networks: the Impact of Width, Depth, and Topology", *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202, pp. 7865–7885, Honolulu, Hawaii, USA, 2023.

26. Chen, M., Z. Wei, Z. Huang, B. Ding and Y. Li, "Simple and Deep Graph Convolutional Networks", *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119, pp. 1725–1735, Virtual, 2020.

27. Lee, S. Y., F. Bu, J. Yoo and K. Shin, "Towards Deep Attention in Graph Neural Networks: Problems and Remedies", *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202, pp. 18774–18795, Honolulu, Hawaii, USA, 2023.

28. Rusch, T. K., B. P. Chamberlain, M. W. Mahoney, M. M. Bronstein and S. Mishra, "Gradient Gating for Deep Multi-Rate Learning on Graphs", *International Con-*

ference on Learning Representations, Kigali, Rwanda, 2023.

29. Chien, E., J. Peng, P. Li and O. Milenkovic, "Adaptive Universal Generalized PageRank Graph Neural Network", *International Conference on Learning Representations*, Virtual, 2021.

30. Liu, M., H. Gao and S. Ji, "Towards Deeper Graph Neural Networks", *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, Virtual Event, CA, USA, 2020.

31. Bo, D., X. Wang, C. Shi and H. Shen, "Beyond Low-frequency Information in Graph Convolutional Networks", *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3950–3957, Virtual, 2021.

32. Karabulut, T. H. and I. M. Baytas, "Channel-Attentive Graph Neural Networks", *IEEE International Conference on Data Mining*, pp. 729–734, Abu Dhabi, UAE, 2024.

33. Dwivedi, V. P., L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu and D. Beaini, "Long Range Graph Benchmark", *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 22326–22340, New Orleans, LA, USA, 2022.

34. Xu, K., W. Hu, J. Leskovec and S. Jegelka, "How Powerful are Graph Neural Networks?", *Proceedings of the 7th International Conference on Learning Representations*, pp. 9104–9120, New Orleans, Louisiana, USA, 2019.

35. Shi, Y., Z. Huang, S. Feng, H. Zhong, W. Wang and Y. Sun, "Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification", *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1548–1554, Virtual, 2021.

36. Luong, T., H. Pham and C. D. Manning, "Effective Approaches to Attention-

based Neural Machine Translation", L. Màrquez, C. Callison-Burch and J. Su (Editors), *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Association for Computational Linguistics, Lisbon, Portugal, 2015.

37. Bahdanau, D., K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", *International Conference on Learning Representations*, San Diego, California, 2015.

38. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need", *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, Long Beach, California, USA, 2017.

39. Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, Doha, Qatar, 2014.

40. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

41. Li, Y., D. Tarlow, M. Brockschmidt and R. Zemel, "Gated Graph Sequence Neural Networks", *International Conference on Learning Representations*, Caribe Hilton, San Juan, Puerto Rico, 2016.

42. Bresson, X. and T. Laurent, "An Experimental Study of Neural Networks for Variable Graphs", *Workshop of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

43. Song, Y., C. Zhou, X. Wang and Z. Lin, "Ordered GNN: Ordering Message Passing to Deal with Heterophily and Over-Smoothing", *International Conference on*

Learning Representations, Kigali, Rwanda, 2023.

44. Zhu, J., Y. Yan, L. Zhao, M. Heimann, L. Akoglu and D. Koutra, "Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs", *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 7793–7804, Vancouver, BC, Canada, 2020.

45. Platonov, O., D. Kuznedelev, M. Diskin, A. Babenko and L. Prokhorenkova, "A Critical Look at the Evaluation of GNNs Under Heterophily: Are We Really Making Progress?", *International Conference on Learning Representations*, Kigali, Rwanda, 2023.

46. Rong, Y., W. Huang, T. Xu and J. Huang, "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification", *Proceedings of the 8th International Conference on Learning Representations*, pp. 2502–2518, Addis Ababa, Ethiopia, 2020.

47. Ying, C., T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen and T.-Y. Liu, "Do Transformers Really Perform Bad for Graph Representation?", *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pp. 28877–28888, Virtual, 2021.

48. Ma, L., C. Lin, D. Lim, A. Romero-Soriano, P. K. Dokania, M. Coates, P. H. Torr and S.-N. Lim, "Graph Inductive Biases in Transformers Without Message Passing", *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202, pp. 23321–23337, Honolulu, Hawaii, USA, 2023.

49. Müller, L., M. Galkin, C. Morris and L. Rampášek, "Attending to Graph Transformers", *Transactions on Machine Learning Research*, 2024.

50. Gasteiger, J., S. Weißenberger and S. Günnemann, "Diffusion Improves Graph Learning", *Proceedings of the 33rd International Conference on Neural Information*

Processing Systems, pp. 13366–13378, Vancouver, Canada, 2019.

51. Topping, J., F. D. Giovanni, B. P. Chamberlain, X. Dong and M. M. Bronstein, "Understanding Over-Squashing and Bottlenecks on Graphs via Curvature", *International Conference on Learning Representations*, Virtual, 2022.

52. Nguyen, K., H. Nong, V. Nguyen, N. Ho, S. Osher and T. Nguyen, "Revisiting Over-Smoothing and Over-Squashing Using Ollivier-Ricci Curvature", *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202, pp. 25956–25979, Honolulu, Hawaii, USA, 2023.

53. Black, M., Z. Wan, A. Nayyeri and Y. Wang, "Understanding Oversquashing in GNNs Through the Lens of Effective Resistance", *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202, pp. 2528–2547, Honolulu, Hawaii, USA, 2023.

54. Banerjee, P. K., K. Karhadkar, Y. G. Wang, U. Alon and G. Montúfar, "Oversquashing in GNNs Through the Lens of Information Contraction and Graph Expansion", *58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1–8, Monticello, IL, USA, 2022.

55. Karhadkar, K., P. K. Banerjee and G. Montufar, "FoSR: First-Order Spectral Rewiring for Addressing Oversquashing in GNNs", *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda, 2023.

56. Gabrielsson, R. B., M. Yurochkin and J. Solomon, "Rewiring with Positional Encodings for Graph Neural Networks", *Transactions on Machine Learning Research*, 2023.

57. Barbero, F., A. Velingker, A. Saberi, M. M. Bronstein and F. D. Giovanni, "Locality-Aware Graph Rewiring in GNNs", *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024.

58. Arnaiz-Rodríguez, A., A. Begga, F. Escolano and N. M. Oliver, "DiffWire: Inductive Graph Rewiring via the Lovász Bound", *Proceedings of the First Learning on Graphs Conference*, Vol. 198, pp. 15:1–15:27, Virtual, 2022.

59. Qian, C., A. Manolache, K. Ahmed, Z. Zeng, G. V. den Broeck, M. Niepert and C. Morris, "Probabilistically Rewired Message-Passing Neural Networks", *The Twelfth International Conference on Learning Representations*, Vienna, Austria, 2024.

60. Deac, A., M. Lackenby and P. Veličković, "Expander Graph Propagation", *Proceedings of the First Learning on Graphs Conference*, Vol. 198, pp. 38:1–38:18, Virtual, 2022.

61. Wilson, J., M. Bechler-Speicher and P. Veličković, "Cayley Graph Propagation", *The Third Learning on Graphs Conference*, Virtual, 2024.

62. Attali, H., D. Buscaldi and N. Pernelle, "Delaunay Graph: Addressing Oversquashing and Over-smoothing Using Delaunay Triangulation", *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235, pp. 1992–2008, Vienna, Austria, 2024.

63. Cai, C., T. S. Hy, R. Yu and Y. Wang, "On the Connection Between MPNN and Graph transformer", *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202, pp. 3408–3430, Honolulu, Hawaii, USA, 2023.

64. Southern, J., F. D. Giovanni, M. M. Bronstein and J. F. Lutzeyer, "Understanding Virtual Nodes: Oversquashing and Node Heterogeneity", *The Thirteenth International Conference on Learning Representations*, Singapore, 2025.

65. Hwang, E., V. Thost, S. S. Dasgupta and T. Ma, "An Analysis of Virtual Nodes in Graph Neural Networks for Link Prediction", *The First Learning on Graphs Conference*, Virtual, 2022.

66. Sestak, F., L. Schneckenreiter, J. Brandstetter, S. Hochreiter, A. Mayr and G. Klambauer, "VN-EGNN: E(3)-Equivariant Graph Neural Networks with Virtual Nodes Enhance Protein Binding Site Identification", ArXiv:2404.07194 [cs.LG], 2024.

67. Zhang, Y., J. Cen, J. Han, Z. Zhang, J. Zhou and W. Huang, "Improving Equivariant Graph Neural Networks on Large Geometric Graphs via Virtual Nodes Learning", *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235, pp. 58662–58679, Vienna, Austria, 2024.

68. Satorras, V. G., E. Hoogeboom and M. Welling, "E(n) Equivariant Graph Neural Networks", *Proceedings of the 38th International Conference on Machine Learning*, Vol. 139, pp. 9323–9332, Virtual, 2021.

69. Choi, J., S. Park, H. Wi, S.-B. Cho and N. Park, "PANDA: Expanded Width-Aware Message Passing Beyond Rewiring", *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235, pp. 8740–8761, Vienna, Austria, 2024.

70. Shao, Z., D. Shi, A. Han, Y. Guo, Q. Zhao and J. Gao, "Unifying Over-Smoothing and Over-Squashing in Graph Neural Networks: A Physics Informed Approach and Beyond", ArXiv:1905.09550 [cs.LG], 2023.

71. Ba, J. L., J. R. Kiros and G. E. Hinton, "Layer Normalization", ArXiv:1607.06450 [stat.ML], 2016.

72. Zhao, L. and L. Akoglu, "PairNorm: Tackling Oversmoothing in GNNs", *Proceedings of the 8th International Conference on Learning Representations*, pp. 4056–4072, Addis Ababa, Ethiopia, 2020.

73. Guo, X., Y. Wang, T. Du and Y. Wang, "ContraNorm: A Contrastive Learning Perspective on Oversmoothing and Beyond", *The Eleventh International Conference on Learning Representations*, Kigali, Rwanda, 2023.

74. Shuman, D. I., S. K. Narang, P. Frossard, A. Ortega and P. Vandergheynst, "The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains", *IEEE Signal Processing Magazine*, Vol. 30, No. 3, pp. 83–98, 2013.

75. Rusch, T. K., B. Chamberlain, J. Rowbottom, S. Mishra and M. Bronstein, "Graph-Coupled Oscillator Networks", *Proceedings of the 39th International Conference on Machine Learning*, Vol. 162, pp. 18888–18909, Baltimore, Maryland, USA, 2022.

76. Finkelshtein, B., X. Huang, M. Bronstein and u. u. Ceylan, "Cooperative Graph Neural Networks", *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235, pp. 13633–13659, Vienna, Austria, 2024.

77. Page, L., S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", Stanford InfoLab 1999-66, 1999.

78. Raghavan, U. N., R. Albert and S. Kumara, "Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks", *Physical Review E*, Vol. 76, No. 3, pp. 036106–1–036106–11, 2007.

79. Cordasco, G. and L. Gargano, "Community Detection via Semi-Synchronous Label Propagation Algorithms", *IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA)*, pp. 1–8, Bangalore, India, 2010.

80. Brandes, U., "A Faster Algorithm for Betweenness Centrality", *The Journal of Mathematical Sociology*, Vol. 25, No. 2, pp. 163–177, 2001.

81. Newman, M. E. J., "Scientific Collaboration Networks. II. Shortest Paths, Weighted Networks, and Centrality", *Physical Review E*, Vol. 64, No. 1, pp. 016132–1–016132–7, 2001.

82. Sen, P., G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher and T. Eliassi-Rad,

"Collective Classification in Network Data", *AI Magazine*, Vol. 29, No. 3, pp. 93–106, 2008.

83. Chen, J., T. Ma and C. Xiao, "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling", *Proceedings of the 6th International Conference on Learning Representations*, pp. 3104–3118, Vancouver, Canada, 2018.

84. Pei, H., B. Wei, K. C.-C. Chang, Y. Lei and B. Yang, "Geom-GCN: Geometric Graph Convolutional Networks", *Proceedings of the 8th International Conference on Learning Representations*, pp. 10247–10258, Addis Ababa, Ethiopia, 2020.

85. Lim, D., F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao and S.-N. Lim, "Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods", *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pp. 20887–20902, Virtual, 2021.

86. Morris, C., N. M. Kriege, F. Bause, K. Kersting, P. Mutzel and M. Neumann, "TUDataset: A Collection of Benchmark Datasets for Learning with Graphs", *ICML Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, Virtual, 2020.

87. Xu, K., C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi and S. Jegelka, "Representation Learning on Graphs with Jumping Knowledge Networks", *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80, pp. 5453–5462, Stockholm, Sweden, 2018.

88. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations*, San Diego, California, 2015.

89. Fey, M. and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric", *ICLR Workshop on Representation Learning on Graphs and Manifolds*, New Orleans, Louisiana, USA, 2019.

90. Rossi, E., B. Charpentier, F. D. Giovanni, F. Frasca, S. Günnemann and M. M. Bronstein, "Edge Directionality Improves Learning on Heterophilic Graphs", *Proceedings of the Second Learning on Graphs Conference*, Vol. 231, pp. 25:1–25:27, Virtual, 2024.

91. Ghosh, A., S. Boyd and A. Saberi, "Minimizing Effective Resistance of a Graph", *SIAM Review*, Vol. 50, No. 1, pp. 37–66, 2008.

92. Hu, W., M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta and J. Leskovec, "Open Graph Benchmark: Datasets for Machine Learning on Graphs", *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 22118–22133, Vancouver, BC, Canada, 2020.

APPENDIX A: VISUALIZATIONS

We present visualizations of pairwise cosine similarities of channel weights for five randomly selected nodes. We limit the number of nodes because of space limitations. We have observed similar results for all nodes. The visualizations for four datasets can be seen in Figures A.1, A.2, A.3. The results are obtained from all layers of the best models, which we report the results of in Tables 4.3, 4.4 and 4.5. For highly heterophilous datasets, we can observe that nodes send different messages to their neighbors. Figures A.1 and A.2 show that, in Minesweeper and Roman-Empire datasets, the message-passing scheme changes with respect to different nodes and layers. This shows that CHAT-GNN leverages the channel-attentive mechanism to differentiate the sent messages between neighbors. We also observe that nodes send almost identical messages to their neighbors in graphs with high homophily. In Figure A.3, we show that the nodes send identical messages to their neighbors in Pubmed. The messages become different in some cases for the second layer.

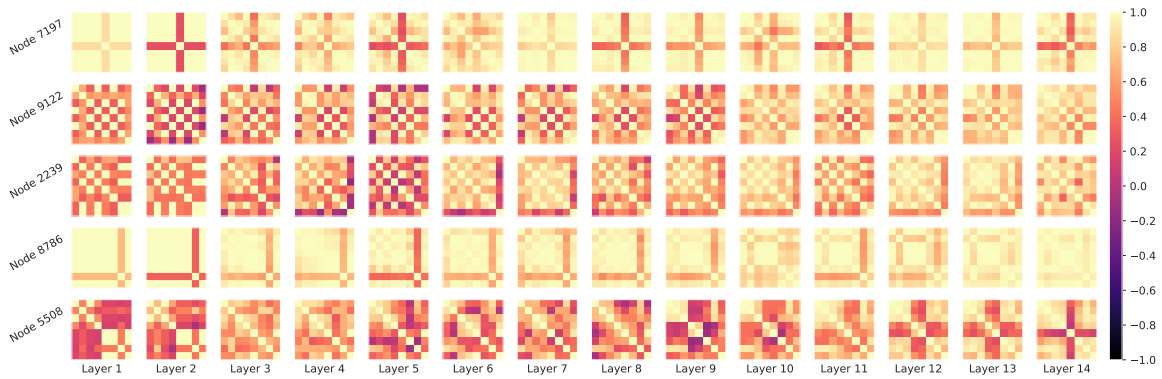


Figure A.1. Visualization of pairwise cosine similarities of channel attention vectors at deeper layers for Minesweeper dataset.

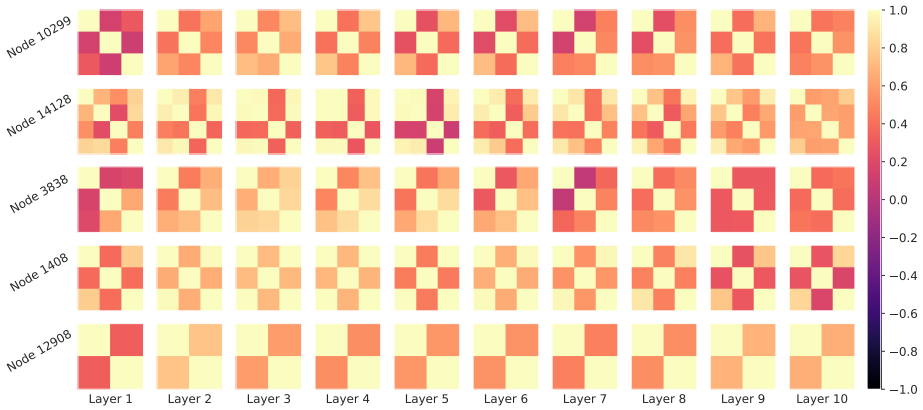


Figure A.2. Visualization of pairwise cosine similarities of channel attention vectors at deeper layers for Roman-Empire dataset.

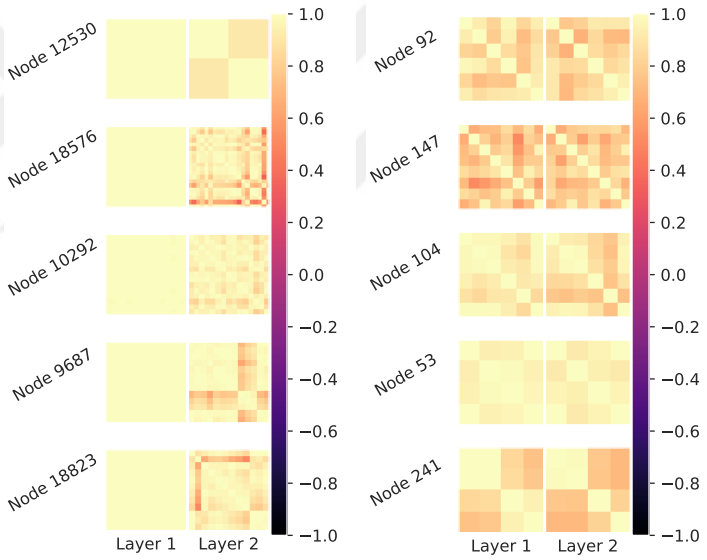
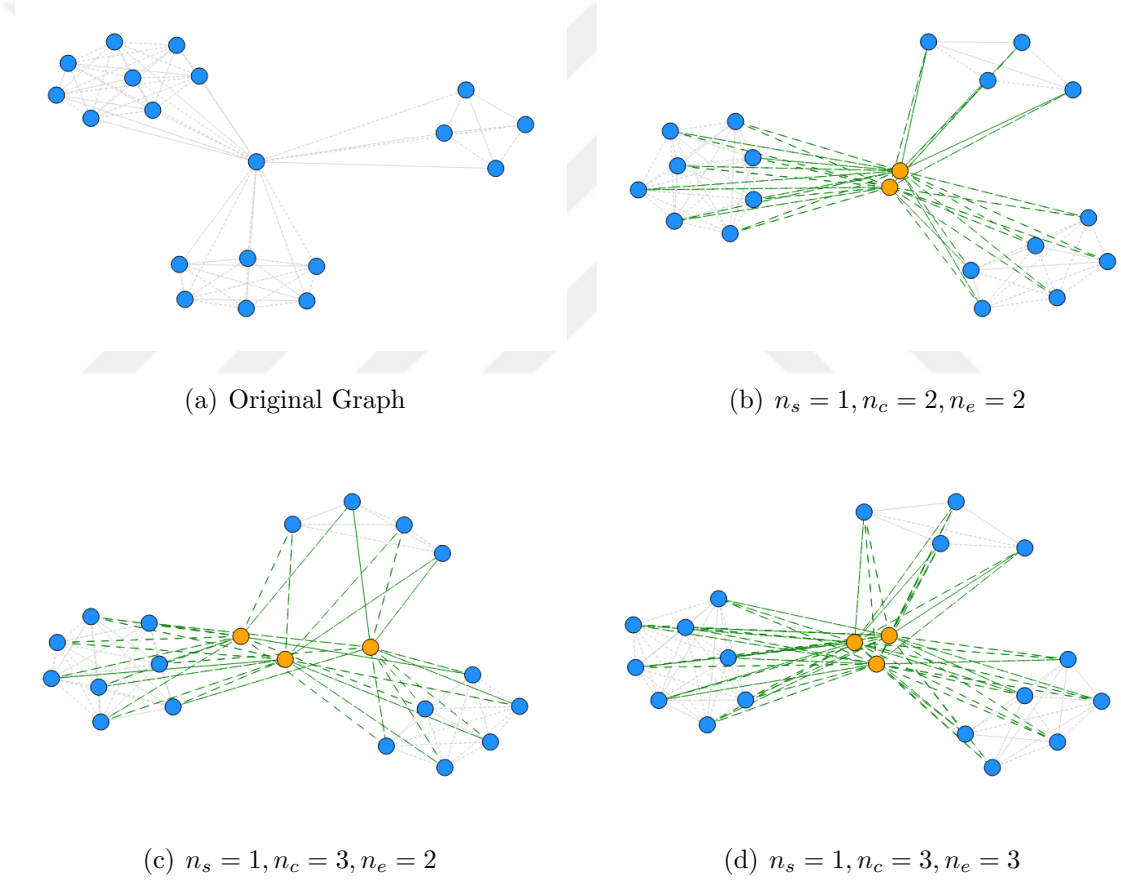


Figure A.3. Visualization of pairwise cosine similarities of channel attention vectors at deeper layers for Pubmed and Wisconsin datasets.

APPENDIX B: GRAPH VISUALIZATIONS

We present illustrated examples for our virtual node integration approach. Figures B.1, B.2 and B.3 show examples of a single graph with and without virtual nodes from datasets IMDB-BINARY, ENZYMES and MUTAG, respectively. Illustrations demonstrate that additional nodes introduce more paths passing through bottlenecks.



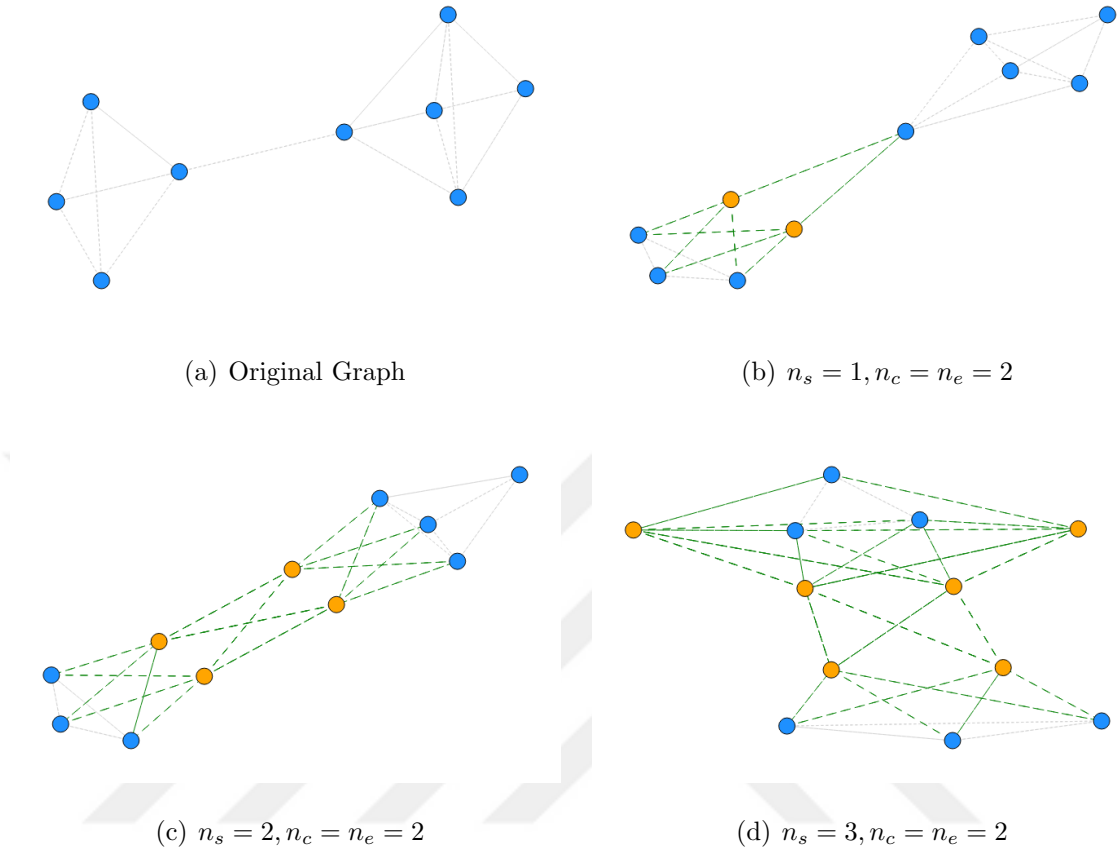
(a) Original Graph

(b) $n_s = 1, n_c = 2, n_e = 2$

(c) $n_s = 1, n_c = 3, n_e = 2$

(d) $n_s = 1, n_c = 3, n_e = 3$

Figure B.1. Visualization of an IMDB-BINARY graph without and with virtual nodes. Blue nodes represent original nodes, while orange nodes indicate virtual nodes added using degree centrality with $n_s = 1$. Dashed green lines represent virtual node connections.



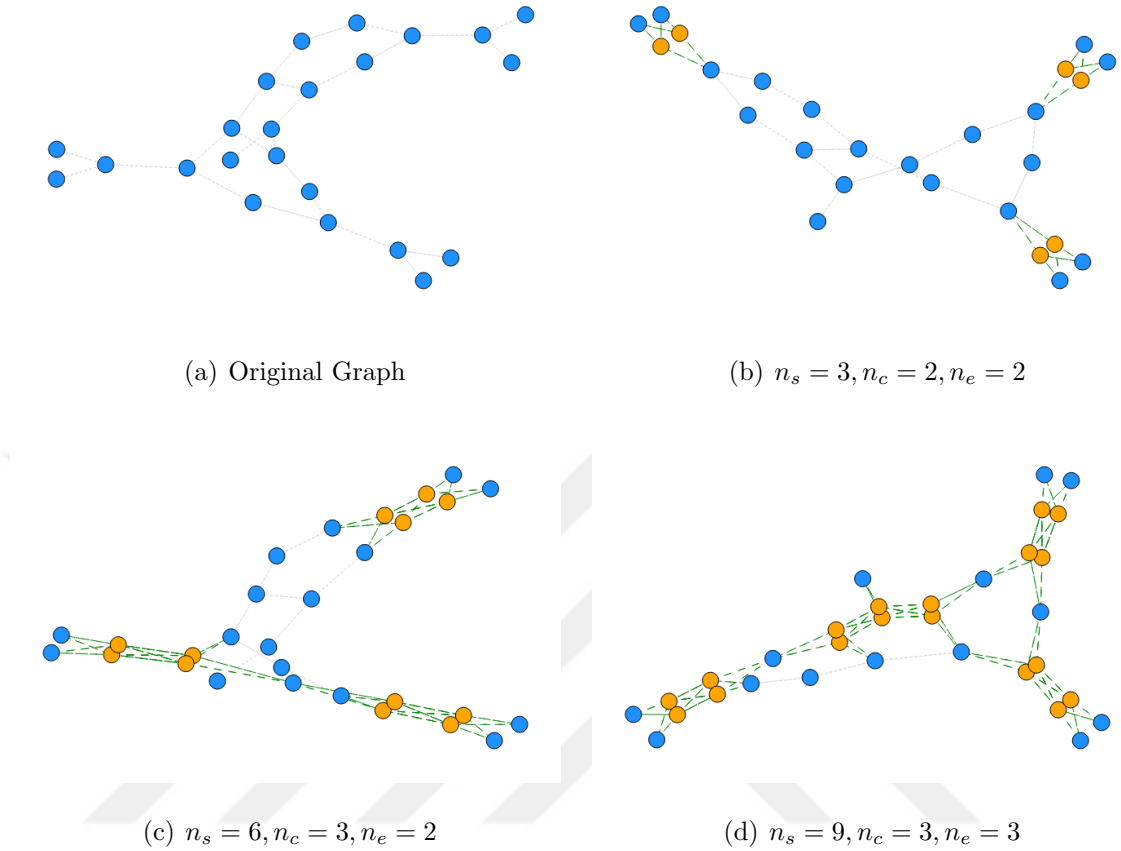
(a) Original Graph

(b) $n_s = 1, n_c = n_e = 2$

(c) $n_s = 2, n_c = n_e = 2$

(d) $n_s = 3, n_c = n_e = 2$

Figure B.2. Visualization of an ENZYMES graph without and with virtual nodes for varying n_s. PageRank is used as the centrality function.



(a) Original Graph

(b) $n_s = 3, n_c = 2, n_e = 2$

(c) $n_s = 6, n_c = 3, n_e = 2$

(d) $n_s = 9, n_c = 3, n_e = 3$

Figure B.3. Visualization of a MUTAG graph without and with virtual nodes. PageRank is the centrality function.