Multivariate Time Series Forecasting By Graph Attention Networks With Theoretical Guarantees

Zhi Zhang
University of California, Los Angeles

Weijian Li
Northwestern University

Han Liu
Northwestern University

Abstract

Multivariate time series forecasting (MTSF) aims to predict future values of multiple variables based on past values of multivariate time series, and has been applied in fields including traffic flow prediction, stock price forecasting, and anomaly detection. Capturing the inter-dependencies among multiple series poses one significant challenge to MTSF. Recent works have considered modeling the correlated series as graph nodes and using graph neural network (GNN)-based approaches with attention mechanisms added to improve the test prediction accuracy, however, none of them have theoretical guarantees regarding the generalization error. In this paper, we develop a new norm-bounded graph attention network (GAT) for MTSF by upper-bounding the Frobenius norm of weights in each layer of the GAT model to enhance performance. We theoretically establish that the generalization error bound for our model is associated with various components of GAT models: the number of attention heads, the maximum number of neighbors, the upper bound of the Frobenius norm of the weight matrix in each layer, and the norm of the input features. Empirically, we investigate the impact of different components of GAT models on the generalization performance of MTSF on real data. Our experiment verifies our theoretical findings. We compare with multiple prior frequently cited graph-based methods for MTSF using real data sets and the experiment results show our method can achieve the best performance for MTSF. Our method provides novel perspectives for improving the generalization performance of MTSF, and our theoretical guarantees give substantial implications for designing graph-based methods with attention mechanisms for MTSF.

Proceedings of the 27[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

1 INTRODUCTION AND BACKGROUNDS

Complex time series data generated in the real world make MTSF a crucial topic in various scenarios, such as traffic forecasting, sensor signal anomaly detection in the Internet of things, demand and supply prediction in the supply chain management, and stock market price prediction in financial investment. Traditional methods like auto-regressive (Mills & Mills, 1990), auto-regressive integrated moving average (Box et al., 2015) and vector auto-regression (Box et al., 2015) have been employed for time series forecasting. In addition to these, deep learning methods (Bai et al., 2018; Zhang et al., 2017; Tokgöz & Ünal, 2018) including graph based neural networks (Yu et al., 2017; Wu et al., 2019; Chen et al., 2020b) have been applied in MTSF problems and demonstrated better performance to solve these problems.

The attention mechanism (Vaswani et al., 2017) is also used along with the GNN by Guo et al. (2019); Deng & Hooi (2021) to adaptively adjust the correlations among multiple time series. One of the seminal contributions to GNN with attention mechanisms is the GAT framework proposed by (Veličković et al., 2017). In GAT, every node computes the importance of its neighboring nodes, and then utilizes the importance as weights to update its representations of the features during the aggregation, it can handle complex inter-dependency (i.e., the correlations) among nodes.

The generalization error bound provides a standard approach to evaluate neural networks as it characterizes the predictive performance of a class of learning models for unseen data (Golowich et al., 2018). Many frameworks specifically designed for MTSF that rely on neural networks due to their high prediction accuracy and ability to handle interdependencies among

different series. Graph-based models are among these approaches. Despite their advantages, addressing generalization error in neural networks remains crucial, as strong performance on training data does not guarantee similar results on unseen data. To capture the correlations among multiple correlated time series, we use the GATs for MTSF, where each node represents a single time series. Moreover, we observe a gap in the theoretical understanding of generalization error of GATs for MTSF, specifically, existing studies on GNNs with attention mechanism lack theoretical guarantees regarding generalization error bounds. Our paper addresses this aspect by developing the theoretical generalization error bound for GATs for MTSF. Our generalization error bound is derived using the Rademacher complexity bounds, a method employed for deep neural networks in several foundational studies by Bartlett (1998); Bartlett & Mendelson (2002); Bartlett et al. (2017); Neyshabur et al. (2015); Golowich et al. (2018); Garg et al. (2020). The generalization error bounds derived in this study for MTSF rely on bounding the Empirical Rademacher Complexity (ERC) of GAT models with the weight matrix norm being bounded. Often this derivation involves controlling the norm of the hidden layer weight matrix, a strategy for establishing norm-based generalization error bounds in DNNs, CNNs, and GNNs. It is significant that Edelman et al. (2022) and Fu et al. (2024) investigated the generalization bounds of a single attention network by analyzing the covering number-based capacity of the function class and employing kernel methods, respectively. In our study, we explore the generalization error bounds for GATs in MTSF through Rademacher complexity. Additionally, we provide both theoretical and experimental evidence to support the control of the weight matrix norm as a means to enhance the generalization capabilities of GATs. The generalization error bound we derived includes different components of GATs models: the number of attention heads, the maximum number of neighbors, the upper bound of the Frobenius norm of the weight matrix in each layer, and the norm of the input features. We investigate the influence of these components on the generalization performance of MTSF using experiments with complex stock price and weather data. Experimental results are consistent with our theoretical findings.

Among all the factors mentioned above that influence the generalization error, the scaling of the upper bound of the weights' norm is particularly significant. Since it can be adjusted, properly controlling the upper bound of the norm of the weight matrix is especially significant. GATs that do not constrain weights may end up with layers that blow-up in the norm. Weights with abnormally large norms may cause training instability and increased generalization error. With this consideration, we propose a new method, Weights-bounded GAT for MTSF. We specify a threshold value based on the the weight matrix size to ensure that no element of the weight matrix exceeds this limit, thereby making sure the Frobenius norm of the weight matrix within the predefined bound throughout the GATs training and testing process. This method resembles the weight norm control strategy seen in Hinton et al. (2012), which also aimed to reduce generalization error by limiting the L_2 norm of the weight matrix. Our method ensures GATs to maintain bounded weights throughout the training process to prevent overfitting. To evaluate the performance of our method, we compare our method with six GNN-based methods for MTSF and show that our method outperforms these methods for MTSF with a smaller generalization error.

Our main **contributions** are as follows:

- We derive the theoretical generalization error bound for GAT for MTSF that provides a theoretical analysis of each GAT component's influence on the prediction performance. We also conduct experiments to demonstrate their influence empirically. Generalization error bounds derived in this study are based on the bound of ERC of GAT models with the weight matrix norm being controlled.
- Based on our theoretical findings on the relations between different GAT components and the generalization error bound, we propose the Weights-bounded GAT. Our experiment results show that it has a smaller generalization loss compared to the vanilla GAT and other GNN-based models on the MTSF task.

2 PRELIMINARIES

2.1 Problem Formulation And The Graph Structure

In this paper, we focus on the task of MTSF, considering a multivariate situation that contains N correlated univariate time series with each represented as $\{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,t}, \ldots\}$ to denote a sequence of time series i from time step 1 to infinity. Based on a sequence of historical T time steps of values prior to current time t for N time series, $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, with each $\mathbf{x}_i = \{\mathbf{x}_{i,t}, \ldots, \mathbf{x}_{i,t-T+1}\}$, with each $\mathbf{x}_{i,t} \in \mathbb{R}^{D_t}$, our goal is to predict the multi-step-away value of $\{\mathbf{y}_1, \ldots, \mathbf{y}_i\}$ using an appropriate prediction model f, where each $\mathbf{y}_i = \{\mathbf{y}_{i,t+1}, \ldots, \mathbf{y}_{i,t+C}\}$ has values from C timestamps, with each $\mathbf{y}_{i,t+c} \in \mathbb{R}, c = 1 \ldots C$. In addition, the historical inputs can be representative of multiple aspects if complemented with auxiliary features, so our problem can be characterized as $\mathbf{y}_i = f(\{\mathbf{x}_{1,t}, \ldots, \mathbf{x}_{1,t-T+1}\}, \ldots, \{\mathbf{x}_{i,t}, \ldots, \mathbf{x}_{i,t-T+1}\})$, for $i = 1, \ldots, N$. To capture the inter-dependency,

the problem is formulated on the graph structure as introduced below.

We consider an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. The $\mathcal{N} = \{1, \ldots, N\}$, $|\mathcal{N}| = N$, is a set of nodes representing the sources of N time series. $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is the set of edges representing the connection between series. We let $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$ be a random variable representing the input feature vector of node i for time series i. For node i, its corresponding input feature $\mathbf{x}_i$ is a multi-dimensional vector, which contains all the historical values from T time steps, in other words, we let $\mathbf{x}_i = (\mathbf{x}_{i,t}, \ldots, \mathbf{x}_{i,t-T+1}) \in \mathbb{R}^D$ be the concatenation of T time steps, such that $D = D_t \cdot T$; its true label $\mathbf{y}_i \in \mathbb{R}^C$ is the vector for the C-step-away values.

2.2 GATs Model

We consider the GATs defined by Veličković et al. (2017), given the random feature matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$, the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a L-layers GAT model is $f : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times C}$, with the final output $\mathbf{Z}_{(L)} \in \mathbb{R}^{N \times C}$ as,

$$\mathbf{Z}_{(L)} = f(\mathbf{X}), \tag{1}$$

$$\begin{aligned} \text{with } f(\mathbf{X}) = \mathbf{P}_{(L)} \oplus_{k=1}^{K} \sigma \big(& \mathbf{P}_{(L-1)} \\ & \cdots \oplus_{k=1}^{K} \sigma \left(\mathbf{P}_{(2)} \oplus_{k=1}^{K} \sigma \left(\mathbf{P}_{(1,k)} \mathbf{X} \mathbf{W}_{(1,k)} \right) \mathbf{W}_{(2)} \right) \\ & \cdots \mathbf{W}_{(L-1)} \big) \mathbf{W}_{(L)} \in \mathbb{R}^{N \times C} \end{aligned} \tag{2}$$

to predict the true label $[\mathbf{y}_1, \mathbf{y}_2 \ldots, \mathbf{y}_N]^\top \in \mathbb{R}^{N \times C}$.

In a GAT with more than two layers, the output of the hidden layer l is

$$\mathbf{Z}_{(l)} = \oplus_{k=1}^{K} \sigma(\mathbf{P}_{(l,k)} \mathbf{Z}_{(l-1)} \mathbf{W}_{(l,k)}) \in \mathbb{R}^{N \times D_l K}, \tag{3}$$

Here we use subscript (l) or $(l-1)$ to indicate which layer the variable belongs to. We have $l \in \{1, 2, \ldots, L-1\}$, and $\mathbf{Z}_0 = \mathbf{X}$.

Here, $\mathbf{W}_{(1,k)} \in \mathbb{R}^{D \times D_1}$, $\mathbf{W}_{(l,k)} \in \mathbb{R}^{K D_{l-1} \times D_l}$, for $l \geq 2$, is an $l-1$-to-l weight matrix for a hidden layer with D_l feature maps. Here σ is the activation function.

And we let $\oplus$ denote the concatenation for the *attention heads*. This definition is specific to the GATs, with its detailed description found in Veličković et al. (2017).

And we have total K such matrices in layer l with each matrix $\mathbf{W}_{(l,k)}$ corresponding to one attention head. $\mathbf{P}_{(1,k)}$, $\mathbf{P}_{(2,k)}$ and $\mathbf{P}_{(L-1,k)}$ are the *attention matrix* introduced by us, and function as an operator to incorporate the attention. We will justify its equivalence to the original GAT models (Veličković et al., 2017) in later paragraphs.

Even though our analysis covers GATs with more than two layers, we will give a focus on two-layers GATs model, which is also implemented by Veličković et al. (2017), with the following simple form:

$$\mathbf{Z}_{(2)} = f(\mathbf{X}) \tag{4}$$

$$f(\mathbf{X}) = \mathbf{P}_{(2)} \oplus_{k=1}^{K} \sigma \left(\mathbf{P}_{(1,k)} \mathbf{X} \mathbf{W}_{(1,k)} \right) \mathbf{W}_{(2)}. \tag{5}$$

There are multiple attention heads in the $1, \ldots, L-1$ layers, but there is only one attention head in the L-th layer. That's why in Equation (5), we use $\mathbf{P}_{(2)}$ instead of $\mathbf{P}_{(2,k)}$ for a two layers neural network. This is consistent with the standard setup for GATs.

The first layer consists of K attention heads computing D_1 features each (a total of $K \times D_1$ features), followed by an activation function σ. Here, $\mathbf{W}_{(1,k)} \in \mathbb{R}^{D \times D_1}$ is an input-to-hidden weight matrix for a hidden layer with D_1 feature maps, and we have K such matrices.

The second layer is used for prediction: a single attention head that predicts the $\mathbf{y}$. For C-step-away forecasting, we have $D_L = C$. The $\mathbf{W}_{(2)} \in \mathbb{R}^{K D_1 \times C}$ is a hidden-to-output weight matrix.

Attention Model We now give more explanation about the *attention* introduced in the GAT model. In section 2 of the original GAT paper, Veličković et al. (2017) mentioned the *learnable linear transformation* to transform the input features into higher-level features with sufficient expressive power. In their process, they first apply a shared linear transformation, parameterized by the weight matrix, $\mathbf{W} \in \mathbb{R}^{D \times D_1}$, to every node. Then they perform *self-attention* on the node: a shared attentional mechanism $a : \mathbb{R}^{D_1} \times \mathbb{R}^{D_1} \to \mathbb{R}$ computes attention coefficients $e_{ij} = a(\mathbf{x}_i^\top \mathbf{W}, \mathbf{x}_j^\top \mathbf{W})$, to indicate the importance of node j's features to node i. Let $\mathcal{N}(i)$ denote the neighborhood of node i (including i). They also inject the graph structure into the mechanism by performing masked attention: they compute e_{ij} for nodes $j \in \mathcal{N}(i)$, the neighbors of node i, which might include node i itself. Then they normalize the coefficients to make them easily comparable across different nodes using the softmax function to obtain $p^{i,j}$

$$p^{i,j} = \phi([e_{i,1}, e_{i,2}, \ldots])^j = \frac{\exp e_{i,j}}{\sum_{j' \in \mathcal{N}(i)} \exp e_{i,j'}}.$$

Then the output features from the first layer for each node will be: $\sigma(\sum_{j \in \mathcal{N}(i)} p^{i,j} \mathbf{x}_j^\top \mathbf{W})$. They also propose the K-head attention. The K independent attention mechanisms execute the aforementioned transformation, and then their output are concatenated, resulting in the following feature representation: $\oplus_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}(i)} p_{(k)}^{i,j} \mathbf{x}_j^\top \mathbf{W}_{(k)})$.

To make the above process more integrated, here in the GAT models, we introduce the attention matrix $\mathbf{P}$ that contains the normalized attention coefficients used

to compute a linear combination of the neighborhood features, yielding the new feature representation for every node. This matrix contains individual node's importance weights with every other node in its neighborhood.

For any $k \in [K]$, let $\mathbf{P}_{(l,k)} \in \mathbb{R}^{N\times N}$, $l \in [L]$, is the matrix of the graph attention matrix defined by the attention coefficients [1]. We define the graph attention matrix $\mathbf{P}_{(l,k)}$ with entries

$$\left[\mathbf{P}_{(l,k)}\right]_{ij} \coloneqq p^{i,j}_{(l,k)} = 0, \quad \text{if } j \notin \mathcal{N}(i). \tag{6}$$

In (6), $p^{i,j}_{(l,k)} \in [0,1]$ is the coefficients of node i attributed from node j. Each row sum of the $\mathbf{P}_{(l,k)}$ is equal to 1, which is $\sum_{j\in\mathcal{N}(i)} p^{i,j}_{(l,k)} = 1$. Here $\mathbf{p}^n_{(l,k)}$ (row i of $\mathbf{P}_{(l,k)}$) denotes node i's all coefficients.

In the paper by Veličković et al. (2017), the proposed attention mechanism a computes the attention coefficients—$e_{i,j}$ and later applies the normalized attention coefficients $p^{i,j}$. We integrate the result of this whole process into an attention matrix $\mathbf{P}_{(l,k)}$, which incorporates the attention. Since the sum of row elements of the $\mathbf{P}$ equals one, we have the following property hold for $\mathbf{P}_{(l,k)}$: $\left\|\mathbf{P}_{(l,k)}\right\|_F \leq N$, i.e., the Forbunis norm of $\mathbf{P}_{(l,k)}$ is bounded by the total number of nodes.

2.3 The Empirical Risk Framework for MTSF

We first introduce function spaces of GATs for MTSF. In our paper, given the feature $\mathbf{X} \in \mathbb{R}^{N\times D}$, and the true label $\mathbf{y} \in \mathbb{R}^C$, we let function class $\mathcal{F}$ be the space of our GAT classes that contains the GAT functions f in (2) which outputs $f(\mathbf{X})$. In other words, we let $\mathcal{F}$ be hypothesis class for functions $f : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times C}$, with f defined in (2) We defer the detailed definitions of $\mathcal{F}$ in later sections, i.e. in terms of Weights-bounded GAT for the MTSF problem, see the Definition 13 and Definition 16.

To predict the true label $\mathbf{y}_i$ for node i where $i = 1,\ldots,N$, we then introduce $\mathcal{I} = \left\{\mathbf{e}_i \in \mathbb{R}^N, i = 1,\ldots,N\right\}$ as the space that contains the $N-$dimensional standard basis vector that selects the index of the node label. Given the node basis vector $\mathbf{e}_i \in \mathcal{I}$, then the node i's output for f is given by $\mathbf{e}_i^\top f(\mathbf{X}) \in \mathbb{R}^C$.

To learn the f, we collect $n \in \mathbb{N}^+$ training data, with the training set $S = \{(\mathbf{X}_1,\mathbf{Y}_1),\ldots,(\mathbf{X}_n,\mathbf{Y}_n)\}$ contains n size of samples, where for each sample $j = 1,\ldots,n$, we have feature $\mathbf{X}_j = \left[\mathbf{x}_{1j},\mathbf{x}_{2j},\ldots,\mathbf{x}_{Nj}\right]^\top \in \mathbb{R}^{N\times D}$, true label $\mathbf{Y}_j = \left[\mathbf{y}_{1j},\mathbf{y}_{2j},\ldots,\mathbf{y}_{Nj}\right]^\top \in \mathbb{R}^{N\times C}$. Then we let $g : \mathbb{R}^C \times \mathbb{R}^C \to [0,1]$ be the loss function. We then introduce the function class $\mathcal{G}_\mathcal{F}$ defined on $\mathcal{I} \times \mathbb{R}^{N\times D} \times \mathbb{R}^C$ by composing the functions in $\mathcal{F}$ with $g(\cdot,\cdot)$, i.e.,

$$\mathcal{G}_\mathcal{F} = \left\{(\mathbf{e}_i,\mathbf{X},\mathbf{y}) \mapsto g(\mathbf{e}_i^\top f(\mathbf{X}),\mathbf{y}); f \in \mathcal{F}\right\}. \tag{7}$$

We assume that g is bounded, i.e., the range of loss is $[0,1]$ (if not, we can scale the loss function) without loss of generality. Additionally, we suppose g is Lipschitz with constant L_g.

For the risk function g in $\mathcal{G}_\mathcal{F}$ defined over $\mathcal{F}$ satisfying the above conditions, given the expected/population risk $\mathbf{E}(g)$ and the empirical risk function $\hat{\mathbf{E}}_n(g)$ are defined as:

$$\mathbf{E}(g) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^N g\left(\mathbf{e}_i^\top f(\mathbf{X}),\mathbf{y}_i\right)\right]. \tag{8}$$

$$\hat{\mathbf{E}}_n(g) = \frac{1}{n}\sum_{j=1}^n \frac{1}{N}\sum_{i=1}^N g\left(f^\top(\mathbf{X}_j)\mathbf{e}_i,\mathbf{y}_{ij}\right). \tag{9}$$

A predictor $f \in \mathcal{F}$ given such $g \in \mathcal{G}_\mathcal{F}$ can be generalized if $\limsup_{|S|=n\to\infty} \hat{\mathbf{E}}_n(g) \to \mathbf{E}(g)$ a.s.

A predictor with a generalization guarantee is to obtain a bound of $\mathbf{E}(g) - \hat{\mathbf{E}}_n(g)$. It is closely related to the complexity of its hypothesis space. In that sense, we would like to obtain a uniform bound:

$$\sup_{g\in\mathcal{G}_\mathcal{F}} \mathbf{E}(g) - \hat{\mathbf{E}}_n(g). \tag{10}$$

for all functions in $\mathcal{G}_\mathcal{F}$. Using empirical process theory, the above error can be bounded by the Rademacher Complexity of the function class $\mathcal{F}$ with high probability, see Theorem 3.2 for an example.

2.4 The Rademacher Complexity

Given a GATs model space $\mathcal{F}$, which is subject to be defined later, and which contains functions $f : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times C}$ in (2), and fixed an $\mathbf{e}_i \in \mathcal{I}$ for $i = 1,\ldots,N$, we introduce the function class $\mathcal{F}_i$ as

$$\mathcal{F}_i = \left\{f_i : \mathbf{X} \mapsto \mathbf{e}_i^\top f(\mathbf{X}); f \in \mathcal{F}\right\}. \tag{11}$$

Then given the training set S and $\mathcal{F}_i$, then, the ERC over $\mathcal{F}_i$ is defined as

$$\mathcal{R}(\mathcal{F}_i) = \mathbb{E}_\epsilon\left[\sup_{f_i\in\mathcal{F}_i} \frac{1}{n}\sum_{j=1}^n \epsilon_j f_i(\mathbf{X}_j)\right]. \tag{12}$$

where $\{\epsilon_1,\cdots,\epsilon_n\}$ are i.i.d. indexed Rademacher sequence satisfying $\mathbb{P}(\epsilon_j = 1) = \mathbb{P}(\epsilon_j = -1) = 1/2$.

3 GENERALIZATION BOUND FOR THE GAT MODEL

3.1 Notation

We use bold-faced letters to denote vectors and capital letters to denote matrices or fixed parameters

[1] In the final layer, we can have $\mathbf{P}_{(L)} \in \mathbb{R}^{M\times N}$, where $M \leq N$, if we only care about a part of the nodes.

(which should be clear from the context). Given a vector $\mathbf{w} \in \mathbb{R}^D$, $\|\mathbf{w}\|$ refers to the Euclidean norm. For a matrix $\mathbf{W}$, $\|\mathbf{W}\|_F$ refers to the Frobenius norm, $\|\mathbf{W}\|_F = \sqrt{\sum_i \sum_j |w^{i,j}|^2}$. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is L-Lipschitz, $L \geq 0$, if $\|f(a) - f(b)\| \leq L \|a - b\|$ for all $a, b \in \mathbb{R}^n$. We use standard big-O notation, with $\mathcal{O}(\cdot)$ hiding constants.

3.2 Function Class of GAT

Given the inputs $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$ as multiple time series with each $\mathbf{x}_i$ as input feature for node i, the class of 2-layer GATs for MTSF f maps $\mathbf{X}$ to the output $f(\mathbf{X})$ that represents a C-step-away prediction expressed in Equation (5). We consider a subset of such class requiring each f with a bounded weights norm, expressed as

$$\begin{aligned} \mathcal{F} = \{ & f : \text{defined in in (5)}; \\ & \left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1, \left\|\mathbf{w}_{(2)}^c\right\| \leq M_2, \Big\}, \end{aligned} \tag{13}$$

for all $c = 1, \ldots, C$. We are interested in this function class because our subsequently derived theory for the bound in (10) is controlled by the ERC within this class. Given the $\mathcal{I} = \{\mathbf{e}_i\}_{n=1}^N$ and the true label $\mathbf{y}_i \in \mathbb{R}^C$, we define f_i to predict $\mathbf{y}_i$ by mapping $\mathbf{X}$ to the output $f_i(\mathbf{X})$, for $i = 1, \ldots, N$. Let $\mathcal{F}_i$ be the hypothesis class for $f_i : \mathbb{R}^{N \times D} \to \mathbb{R}^C$ for node i:

$$\mathcal{F}_i = \left\{ \mathbf{X} \mapsto \mathbf{e}_i^\top f(\mathbf{X}); f \in \mathcal{F} \right\}. \tag{14}$$

Furthermore, we also provide a model space $\mathcal{F}_i^c$ for functions $f_i^c : \mathbb{R}^{N \times D} \to \mathbb{R}$ with a single dimensional output that corresponds to the c-th component of model output from $f_i(\mathbf{X})$ for the c-th time step, expressed as

$$\mathcal{F}_i^c = \left\{ \mathbf{X} \mapsto f_i^c(\mathbf{X}); f_i = (f_i^1, \ldots, f_i^c) \in \mathcal{F}_i \right\}. \tag{15}$$

The function classes $\mathcal{F}_i$ and $\mathcal{F}_i^c$ are introduced to facilitate the derivation of our theory, as illustrated in the proof of Theorem 3.2.

3.3 An Upper Bound of Rademacher Complexity of GAT Class

Here we first provide an upper bound of ERC of GAT class $\mathcal{F}_i^c$ for single dimensional output of MTSF.

Theorem 3.1 (Upper Bound of ERC of GAT class $\mathcal{F}_i^c$ for MTSF)**.** *Let the GAT class $\mathcal{F}_i^c$ follows the definition 15, with the activation function $\sigma(\cdot)$ be 1-Lipschitz continuous, and also satisfy $\sigma(0) = 0$ and $\sigma(\alpha z) = \alpha\sigma(z)$ for all $\alpha \geq 0$. Assume that the Frobenius norm of every weights matrix in the first layer of the GAT class is bounded, namely, $\left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1$ with some constant $M_1 > 0$ for $k = 1, \ldots, K$. Also, the norm of the weights vector of the second layer of the GATs is bounded, $\left\|\mathbf{w}_{(2)}^c\right\| \leq M_2$, where $c \in [C]$, with some constant $M_2 > 0$. Let $\mathcal{N}(i)$ denote the neighborhood of node i (including i), also let the constant $N_e \in \mathbb{N}^+$ be defined as the maximum number of neighbors, $N_e := \max_i |\mathcal{N}(i)|$, for all node $i \in \mathcal{N}$.*

Assume the feature vectors $\mathbf{x}_i, i = 1, \ldots N$ for time series come from a bounded domain, namely, that the L_2-norm of $\mathbf{x}_i$ is bounded by some positive constant B: $\{\mathbf{x}_i : \|\mathbf{x}_i\| \leq B\}$. Then let $\mathcal{R}(\mathcal{F}_i^c)$ be the ERC defined in the definition 2.4 for GAT class $\mathcal{F}_i^c$ in the definition 15, given the $\mathcal{I} = \{\mathbf{e}_i\}_{i=1}^N$, and the total n sized training set $S = \{(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)\}$, such that for each sample $j = 1, \ldots, n$, we have $\mathbf{X}_j = [\mathbf{x}_{1j}, \mathbf{x}_{2j}, \ldots, \mathbf{x}_{Nj}]^\top$, $\mathbf{Y}_j = [\mathbf{y}_{1j}, \mathbf{y}_{2j}, \ldots, \mathbf{y}_{Nj}]^\top$, then for all $i = 1, \ldots, N$ and all $c = 1, \ldots, C$, if we let $M_a \coloneqq BK(N_e)^2 M_1 M_2$ then we have

$$\mathcal{R}(\mathcal{F}_i^c) = \mathcal{O}\left(M_a n^{-1/2}\right).$$

We see that this ERC bound has a polynomial dependence on the N_e, the maximum number of neighbors. The worst case is that $N_e = N$, the total number of nodes, in the fully connected graph. A small $N_e < N$ can result in a smaller bound. The proof details can be found in §A.

3.4 Generalization Error Bound of the GAT Class for MTSF

In this section, we will give the final generalization error bound of the GAT class for MTSF. The formal result is in the following theorem and the proof is in §B.

Theorem 3.2. *Define the hypothesis class $\mathcal{G}_\mathcal{F}$ as the definition 7, with $\mathcal{F}$ defined in Definition 13. Let all conditions in Theorem 3.1 be satisfied. We suppose $g : \mathbb{R}^C \times \mathbb{R}^C \to [0, 1]$ is Lipschitz with constant L_g. Then, for any $\delta \in (0, 1)$, for all $f \in \mathcal{F}$ with $g \in \mathcal{G}_\mathcal{F}$, denote $M_b \coloneqq M_a + \sqrt{\ln(2/\delta)}$, then with probability at least $1 - \delta$, we have*

$$\sup_{g \in \mathcal{G}_\mathcal{F}} \mathbf{E}(g) - \hat{\mathbf{E}}_n(g) = \mathcal{O}\left(M_b n^{-1/2}\right),$$

The proof details can be found in §B.

4 EXTENSION TO GAT CLASS WITH LAYERS $L > 2$

Now we extend the analysis to GATs with more than two layers for MTSF and provide generalization error bounds. Here, the proof is done by a simple induction argument using the "peeling-off" technique employed for Rademacher complexity bounds for neural networks. The output of a L-layer GATs represents a multi-C-step-away prediction shown in Expression (2).

Without lose of clarity, here we abuse the notation from 2-layer GATs, and use $\mathcal{F}$ to denote the function class according to the definition of the $L-$layer GATs network. Furthermore, we require that the Frobenius norm of every weight matrix in every layer of the GAT class is bounded, namely, for any $l \in [L]$, $\left\|\mathbf{W}_{(l,k)}\right\|_F \leq M_l$ with some constant $M_l > 0$ for $k = 1, \ldots, K$ and $l = 2, \ldots, L$.

$$\begin{aligned}\mathcal{F} = \{f : &\text{ defined in in (2);} \\ &\left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1, \ldots, \left\|\mathbf{w}^c_{(L)}\right\| \leq M_L\Big\},\end{aligned} \tag{16}$$

for all $c = 1, \ldots, C$. Also, let $\mathcal{F}_i$ be the hypothesis class for $f_i : \mathbb{R}^{N\times D} \to \mathbb{R}^C$ for node i:

$$\mathcal{F}_i = \left\{\mathbf{X} \mapsto \mathbf{e}_i^\top f(\mathbf{X}); f \in \mathcal{F}\right\}. \tag{17}$$

Also, we have single dimensional GAT function space defined as

$$\mathcal{F}_i^c = \left\{\mathbf{X} \mapsto f_i^c(\mathbf{X}); f_i = (f_i^1, \ldots, f_i^c) \in \mathcal{F}_i\right\}. \tag{18}$$

Also, the output up to layer l, $l \in [L-1]$ is given as

$$\begin{aligned}f_{(l)}(\mathbf{X}) = \oplus_{k_l=1}^K \sigma\left(\mathbf{P}_{(l)} \cdots \oplus_{k_2=1}^K \sigma\left(\mathbf{P}_{(2)} \oplus_{k_1=1}^K\right.\right. \\ \left.\left.\sigma\left(\mathbf{P}_{(1,k_1)}\mathbf{X}\mathbf{W}_{(1,k_1)}\right)\mathbf{W}_{(2,k_2)}\right)\cdots \mathbf{W}_{(l,k_l)}\right) \in \mathbb{R}^{N\times D_l K}.\end{aligned} \tag{19}$$

Thus, we define a layer-wised class of functions as

$$\mathcal{F}_{i(l)} = \left\{\mathbf{X} \mapsto \mathbf{e}_i^\top f_{(l)}(\mathbf{X});\ \left\{\left\|\mathbf{W}_{(l',k)}\right\|_F \leq M_{l'}\right\}_{l'\in[l]}\right\}. \tag{20}$$

We provide an upper bound of ERC of GAT class $\mathcal{F}_i^c$ with L layers with proof details in §C

Theorem 4.1 (Upper Bound of ERC of GAT class $\mathcal{F}_i^c$ with L layers)**.** *Let all the assumptions from Theorem 3.1 be fulfilled. Furthermore, let the Frobenius norm of every weight matrix in the first $L-1$ layers of the GATs be bounded, namely, $\left\|\mathbf{W}_{(l,k)}\right\|_F \leq M_l$ with some constant $M_l > 0$ for $k = 1, \ldots, K$ and $l = 1, \ldots, L-1$. Also, the norm of the weight vector of the last layer is also bounded, $\left\|\mathbf{w}^c_{(L)}\right\| \leq M_L$, where $c \in [C]$, with some constant $M_L > 0$. Let $\mathcal{R}(\mathcal{F}_i^c)$ be the ERC defined in Equation 2.4 for GAT class $\mathcal{F}_i^c$ in the definition 16, given the n sized input set, then for all $i = 1, \ldots, N$ and all $c = 1, \ldots, C$, if we let $M_c \coloneqq N_e^L M_L \cdots M_1 K^{L-1}(L-1)^{1/2}B$ then we have*

$$\mathcal{R}(\mathcal{F}_i^c) = \mathcal{O}\left(M_c n^{-1/2}\right).$$

Based on the theorem 4.1, we provide the generalization error bounds for the GAT class with more than two layers in the following theorem.

Theorem 4.2 (Generalization Error Bounds for the GAT Class with More than Two Layers)**.** *Define the hypothesis class $\mathcal{G}_\mathcal{F}$ as the definition 7, with $\mathcal{F}$ defined in Definition 16. Let all conditions in Theorem 4.1 be satisfied. We suppose $g : \mathbb{R}^C \times \mathbb{R}^C \to [0,1]$ is Lipschitz with constant L_g. Then, for any $\delta \in (0,1)$, for all $f \in \mathcal{F}$ with $g \in \mathcal{G}_\mathcal{F}$, which contains L-layer GATs, denote $M_d \coloneqq M_c + \sqrt{\ln(2/\delta)}$, with probability at least $1-\delta$, we have*

$$\sup_{f\in\mathcal{F}} \mathbf{E}(g) - \hat{\mathbf{E}}_n(g) = \mathcal{O}\left(M_d n^{-1/2}\right).$$

In our proof, to give the generalization error bound of a deep GAT, we first derive its ERC. To bound the ERC, we apply the layer-peeling strategy that informally the ERC of L-layer networks is expressed by a factor multiplied by the ERC of $L-1$-layer networks. This factor contains the product of the matrix Frobenius norm of weight matrices in the current layer, the attention head size K, and the maximum number of neighbors N_e in the graph. Inductively, our current bounds scale with the product of these quantities as the network depth increases. Specifically, the norm of the weight matrices, the number of attention heads, and the number of attention neighbors connections contribute as polynomial terms, where the polynomial order roughly matches the layer count L. The bound has an exponential dependence on the network depth.

The generalization error bound in Theorem 4.2 implies that the following attempts can be taken to reduce the generalization error: i) increase the training samples, ii) minimize the empirical loss, and iii) design the neural network carefully to achieve a proper hypothesis class. Increasing the complexity of the hypothesis class can decrease the approximation error but also increase the estimation error due to a large ERC, which leads to undesired test performance in a practical task. In the next section, we will empirically show how structural components of GATs related to complexity could affect the test performance for a MTSF task, providing empirical support for our theoretical findings.

4.1 Verify The Theoretical Bounds By Experiments

Our goal is to show the relationship between the upper bound of the generalization error of the GAT model and variables in the ERC, including the number of attention heads, the maximum number of neighbors, the Frobenius norm of model weights, the norm of inputs, and the number of samples in our training set. We use two multivariate times data: the daily stock price data from Nasdaq and NYSE and Beijing PM2.5 dataset to do predictions. Details about how we use these data are in §E.2. The experiment results in top

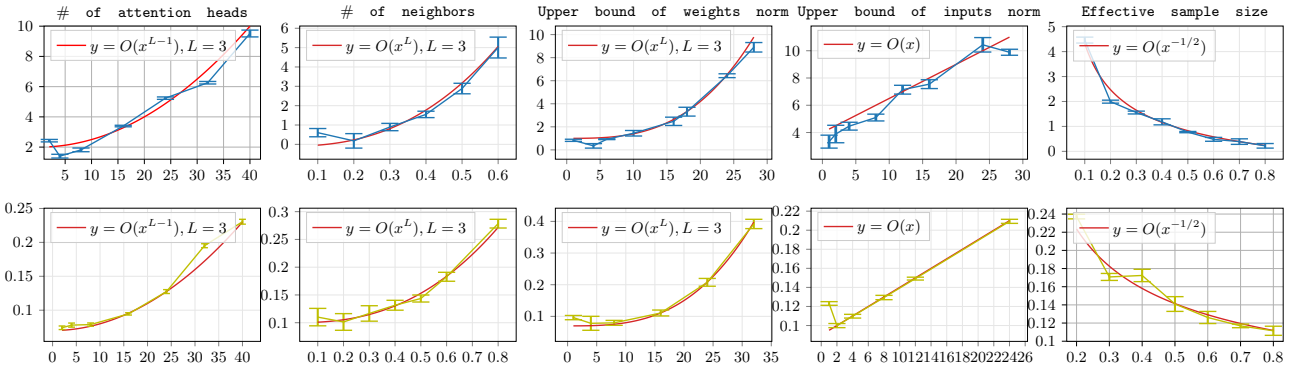

Figure 1: Experiment Results on Five Variables in the ERC. The red line is a possible theoretical upper bound.

row of Figure 1 are about the stock dataset, and the bottom are about the Beijing PM2.5 dataset.

We use a three-layer GAT to demonstrate our results. We include the details about how we control the above six variables in §E.3. In addition, we conduct experiments on two-layer GATs. Both experiments for three and two-layer GATs show that all test losses generally conform to the $\mathcal{O}$ of the theoretical upper bound. error bounds in Theorem 3.1 and 4.1. Results and discussions are in §F.

It is noteworthy that Figure 1 also shows that the test error decreases initially, then starts to increase. The inconsistency observed at at the beginning may be attributed to various factors that impact the test error. As seen in the Theorem 4.2, in addition to the ERC, the empirical risk also contributes to the upper bound of the population risk or generalization error. Nevertheless, the ERC becomes dominant by increasing the number of attention heads, the maximum number of neighbors, and the upper bound of weights norm, indicating the importance of a proper design of neural networks for MTSF to guarantee a smaller generalization error. Among all the factors mentioned above that influence the generalization error, the scaling of the upper bound of the weights' norm is particularly significant, for which we provide justifications subsequently. Since it can be adjusted, properly controlling the upper bound of the weights' norm is considered crucial for developing our method.

As reported in the previous literatures, increasing the complexity hypothesis class in terms of larger weight matrices bounds could decrease the approximation error, but may also increase the estimation error or the generalization error. Therefore, in practical training process, we generally start with a simple neural network and gradually increase its complexity in terms of larger weight matrix bounds to improve the test performance, and the bound can be a tuning parameter in our model. We call it weight control.

5 WEIGHTS-BOUNDED GAT

To show the meanings of weight control, we further developed an improved version of the GAT called Weights-bounded GAT. A three-layer model is described in Appendix §E.3. As the name suggests, the weight Frobenius norm of each layer is bounded by a hyperparameter. Specifically, for the weight norm variable, we adjust the bound of the Frobenius norm of the weight matrix inside each layer by using weight clipping before each forward pass of the model. Each element of the weight matrix is clipped to the threshold to make sure the Frobenius norm of the matrix is less than or equal to the bound. We incorporate weight constraints within the learning process. Specifically, consider a weight matrix $\mathbf{W}_{(1,k)} \in \mathbb{R}^{D \times D_1}$ in (16). During training and testing, before each forward pass, we ensure that every element of the weight matrix adheres to the constraint $-\frac{M_1}{\sqrt{D \times D_1}} \leq \left[\mathbf{W}_{(1,k)}\right]_{dd_1} \leq \frac{M_1}{\sqrt{D \times D_1}}$, with $d = 1, \ldots, D, d_1 = 1, \ldots, D_1$, where M_1 is the upper bound given in our Theorem 3.1. This approach guarantees that the Frobenius norm of the weight matrix remains at or below the specified threshold throughout the training of GAT.

Our proposed approach is akin to regularizing the weight norms in the network to prevent overfitting, similar to other regularization methods. In contrast, commonly used optimizers like SGD and Adam(W) incorporate the weight decay parameter into training setups for (graph attention) neural networks to control the weight norms of the neural network. Although this approach and our method may seem intuitively similar in the sense that they both aim to control the weights of the model, the methodologies they employ to achieve this purpose differ. The weight decay encourages but does not mandate, GAT to possess small weights. On the other hand, weight bounding explicitly controls weight norms during the learning process,

yielding results differing from the implicit weight decay. It compels GAT to maintain bounded weights throughout the training process and during the test period. Given their different approaches to addressing the generalization error, they also exhibit varying theoretical derivations related to generalization error.

To test these two methods empirically, we conducted an experiment to compare the Weights-bounded GAT with its weight norm constraint hyperparameter fine-tuned, against the Vanilla GAT with the weight decay hyperparameter fine-tuned (which results in an implicit weight norm constraint). For hyperparameter tuning, we employed the Bayesian optimization method using Hyperopt (Bergstra et al., 2013). The result is presented in §G, and it sufficiently supports the advantages of our Weights-bounded GAT over a standard GAT with properly tuned weight decay.

To help GAT learn temporal information for each node, we concatenate feature vectors from all historical time steps and subsequently apply a linear transformation. The application of a linear transformation following concatenation can implicitly capture the temporal relationships present in the data. Furthermore, it is worth noting that this approach has also been employed in several instances within the literature on time series forecasting. Specifically, recent studies such as Das et al. (2023); Zeng et al. (2023), utilized a similar approach to process multivariate time series data and have demonstrated exceptional forecasting performance.

5.1 Comparison With Graph-Based Methods

We compare the Weights-bounded GAT with the SOTA graph-based methods using test loss on the above stock return forecasting task. So far, three works ASTGCN (Guo et al., 2019), GMAN (Zheng et al., 2020), and GDN (Deng & Hooi, 2021) use GAT-based models to model multiple time series data, showing better accuracy over other traditional linear methods (e.g., VAR), neural-network based methods (e.g., LSTM), and graph-network based methods (e.g., GNNs and GCNs). While other works in multivariate time series forecasting do not consider the attention mechanism, several frequently cited ones are used as baselines for comparison with Weights-bounded GAT, including STGCN by Yu et al. (2017) and MTGNN by Wu et al. (2020).

All of these works differ from ours as they do not consider any control of model variables (e.g., the weight matrix norm) and they lack theoretical guarantees in terms of the generalization error.

We use two evaluation metrics, the MSE and MAE. Additionally, Table 1 reports their average test error from 20 runs over 20 random seeds. Detailed discussions can also be found in §I. The Weights-bounded GAT has a better test loss than the other baselines.

5.2 Quantile Regression For MTSF

In this section, we also considered the probabilistic forecasting, as it is also widely used in time series forecasting. Specifically, we use quantile regression for time series forecasting. In contrast to the mean regression, the quantile regression has been considered robust thus with its broad application in different fields (Cai, 2002; Davino et al., 2013; Belloni et al., 2023; Zhang et al., 2023). To obtain the estimator, a quantile loss is proposed by Wen et al. (2017)

$$QL(y, \hat{y}, Q) = Q(y - \hat{y})_+ + (1 - Q)(\hat{y} - y)_+, \quad (21)$$

where Q is the cumulative density value. When $Q = 0.5$, the quantile loss QL is simply the Mean Absolute Error, and its minimizer is the median of the predictive distribution, and the MAE is more robust to the outliers compared to the MSE. The larger value of Q (such as $Q = 0.9$) will penalize more on the overestimate, while the smaller value will penalize more on the underestimate. Let $\mathcal{Q} = \{Q^{(l)}\}_{l=1}^{q}$ is the set of quantiles we are interested, then our models are trained to minimize the total loss $\sum_{l=1}^{q} QL(y, \hat{y}^{(l)}, Q^{(l)})$.

The quantile loss function used in this approach models the entire probability distribution of the forecast, as opposed to just a single point estimate. This is crucial in real-world time series forecasting applications where it is necessary to have an understanding of the uncertainty associated with the forecast. Additionally, the quantile loss function enables the model to learn different levels of quantiles, which is useful when the cost of overestimating or underestimating the forecast is not symmetric.

We report the quantile losses for $Q = 0.1$, $Q = 0.5$, and $Q = 0.9$ on both datasets. Table 2 shows that the Weights-bounded GAT method has the smallest quantile loss among all the methods. Specifically, the Weights-bounded GAT has a better test error than the other baselines regarding the 0.1, 0.9 and 0.5 quantile loss. This further validates that our model can achieve the best performance across the entire distribution of the response variable.

5.3 Model Size

To ensure a fair comparison, we also included the number of parameters used by the baseline models in our experiments in Table 3. From the table, we observe that models that operate on both spatial and temporal dimensions (STGCN, MTGNN, ASTGCN), as well as models that consider hierarchical graph structures (MTGNN) or multi-attention mechanisms (GMAN), require a large number of parameters. This increases their risk of overfitting, particularly when many of their

Table 1: MSE MAE (10^{-3})

	US daily stock	
	MSE	MAE
Weight-bounded GAT	**2.04**	**30.61**
ASTGCN (Guo et al. (2019))	7.14	36.77
Vanilla-GAT (Veličković et al. (2017))	7.36	45.91
GDN (Deng & Hooi (2021))	8.11	46.31
GMAN (Zheng et al. (2020))	7.57	38.92
MTGNN (Wu et al. (2020))	9.77	38.77
STGCN (Yu et al. (2017))	7.20	39.94

Table 2: Quantile Loss (10^{-2})

	US daily stock			Beijing PM2.5		
	$Q(0.1)$	$Q(0.5)$	$Q(0.9)$	$Q(0.1)$	$Q(0.5)$	$Q(0.9)$
Weight-bounded GAT	**2.92**	**4.85**	**1.17**	**8.16**	**10.57**	**9.48**
ASTGCN (Guo et al. (2019))	4.07	6.88	1.73	11.38	14.99	14.02
Vanilla-GAT (Veličković et al. (2017))	4.69	7.66	1.87	13.10	16.69	15.11
GDN (Deng & Hooi (2021))	4.89	8.21	2.03	13.69	17.88	16.43
GMAN (Zheng et al. (2020))	4.32	7.21	1.84	12.08	15.71	14.91
MTGNN (Wu et al. (2020))	4.40	7.49	1.87	12.29	16.32	15.11
STGCN (Yu et al. (2017))	4.51	7.81	1.90	12.60	17.01	15.39

Table 3: Model Size

Of Parameters
65417
1390441
65417
32700
209923
286019
244609

weight matrix norm bounds are large. Furthermore, the attention-based graph neural network methods (Deng & Hooi, 2021; Guo et al., 2019; Zheng et al., 2020) may also be susceptible to overfitting, particularly when there is a large number of attention heads or the graph size is large. Our analysis indicates that the generalization error increases with the upper bound of the weight matrix norm, the number of attention heads, and the maximum number of neighbors. As the Tables 1, 2 and 3 together show, our proposed Weights-bounded GAT can achieve the best performance with a generally smaller model size. This proves that the prediction performance advantage of our model does not come from naively scaling up the model size. It further corroborated the weight-bounded control in our method has a significant contribution in boosting up the prediction performance.

The best hyperparameters of Weights-bounded GAT and computational complexity are presented in §E.5 and §E.6.

6 DISCUSSION

We provide future research directions here. First, we discuss the fundamental over-smoothing issue in deep GNN-based models and how our Weights-bounded method could help to mitigate it so that deep GNN-based models can achieve better generalization performance in time series prediction tasks, which leaves us as the future work. Over-smoothing occurs when repeated rounds of message passing lead to node features converging to a non-informative limit. This gradual attenuation of feature significance harms GNN-based models ability to capture the correlation between multiple time series. As a result, the node features become increasingly similar, mixing different data points together, which makes it difficult to achieve precise predictions. Thus, deep GNN-based models today are more likely to suffer from the over-smoothing issue. Works such as Li et al. (2018) firstly have demonstrated that graph over-smoothing arises from repeated applications of Laplacian smoothing. However, GAT's over-smoothing is more intricate due to its multiplication of different attention matrices at different times, adding complexity to the issue. Unlike other GNN based models, where over-smoothing is clearer due to simple aggregation problems, GAT's over-smoothing is less straightforward.

Various researchers have studied the over-smoothing problem in deep GNN-based models and proposed some improvement methods (Li et al., 2018; Oono & Suzuki, 2019; Rong et al., 2019; Chen et al., 2020a; Zhao & Akoglu, 2019; Keriven, 2022; Wu et al., 2022).

Inspired by the above-mentioned works, one of the future research is to investigate more advanced methods for mitigating over-smoothing while preserving generalization guarantees in GATs. For example, the weight-norm bound parameter could be made adaptive; that is, the adaptive weight-norm bound parameters could vary for each layer and they could be applied to exert control differently across different graph nodes. In weight-bound control, the trainable adaptive parameters could potentially mitigate over-smoothing by diversifying the attention matrices across layers and nodes. More diversified attention matrices across layers and nodes – thus, a less similar message passing paths in each layer and for each node– may help mitigate the over-smoothing issue.

Secondly, given the availability of advanced techniques for analyzing generalization error bounds in single attention networks, we believe applying these techniques to GATs could refine and improve current bounds further.

Finally, our bound exhibits polynomial dependence on the maximum number of neighbors. It would be interesting to extend our work to large graphs with a greater number of neighbors. Future research could aim to improve this bound.

Acknowledgements

We would like to express our sincere gratitude to Professor Arash A. Amini from UCLA for his invaluable assistance with this work. His insights and expertise have been instrumental in the completion of this work.

References

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Bartlett, P. L. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Trans. Inf. Theory*, 44(2):525–536, 1998. URL `http://dblp.uni-trier.de/db/journals/tit/tit44.html#Bartlett98`.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3 (Nov):463–482, 2002.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

Belloni, A., Chen, M., Madrid Padilla, O. H., and Wang, Z. High-dimensional latent panel quantile regression with an application to asset pricing. *The Annals of Statistics*, 51(1):96–121, 2023.

Bergstra, J., Yamins, D., Cox, D. D., et al. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, volume 13, pp. 20. Citeseer, 2013.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Cai, Z. Regression quantiles for time series. *Econometric theory*, 18(1):169–192, 2002.

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

Chen, W., Chen, L., Xie, Y., Cao, W., Gao, Y., and Feng, X. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3529–3536, 2020b.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Das, A., Kong, W., Leach, A., Sen, R., and Yu, R. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023.

Davino, C., Furno, M., and Vistocco, D. *Quantile regression: theory and applications*, volume 988. John Wiley & Sons, 2013.

Deng, A. and Hooi, B. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4027–4035, 2021.

Duchi, J. Probability bounds. *URL: http://www. cs. berkeley. edu/jduchi/projects/probability bounds. pdf*, 2009.

Edelman, B. L., Goel, S., Kakade, S., and Zhang, C. Inductive biases and variable creation in self-attention mechanisms. In *International Conference on Machine Learning*, pp. 5793–5831. PMLR, 2022.

Fu, H., Guo, T., Bai, Y., and Mei, S. What can a single attention layer learn? a study through the random features lens. *Advances in Neural Information Processing Systems*, 36, 2024.

Garg, V., Jegelka, S., and Jaakkola, T. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*, pp. 3419–3430. PMLR, 2020.

Golowich, N., Rakhlin, A., and Shamir, O. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pp. 297–299. PMLR, 2018.

Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 922–929, 2019.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Keriven, N. Not too little, not too much: a theoretical analysis of graph (over) smoothing. *Advances in Neural Information Processing Systems*, 35:2268–2281, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ledoux, M. and Talagrand, M. *Probability in Banach Spaces: isoperimetry and processes*, volume 23. Springer Science & Business Media, 1991.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Maurer, A. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory*, pp. 3–17. Springer, 2016.

Mills, T. C. and Mills, T. C. *Time series techniques for economists*. Cambridge University Press, 1990.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

Neyshabur, B., Tomioka, R., and Srebro, N. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401. PMLR, 2015.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

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

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tokgöz, A. and Ünal, G. A rnn based time series approach for forecasting turkish electricity load. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4. IEEE, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wainwright, M. J. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

Wu, X., Chen, Z., Wang, W., and Jadbabaie, A. A non-asymptotic analysis of oversmoothing in graph neural networks. *arXiv preprint arXiv:2212.10701*, 2022.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., and Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 753–763, 2020.

Yu, B., Yin, H., and Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Zhang, L., Aggarwal, C., and Qi, G.-J. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2141–2149, 2017.

Zhang, S., Wang, M., Liu, S., Chen, P.-Y., and Xiong, J. Fast learning of graph neural networks with guaranteed generalizability: one-hidden-layer case. In *International Conference on Machine Learning*, pp. 11268–11277. PMLR, 2020.

Zhang, Z., Ritscher, K., and Padilla, O. H. M. Risk bounds for quantile additive trend filtering. *arXiv preprint arXiv:2310.11711*, 2023.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*, 2019.

Zheng, C., Fan, X., Wang, C., and Qi, J. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 1234–1241, 2020.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [We will provide the code link if accepted.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary Material for Multivariate Time Series Forecasting By Graph Attention Networks With Theoretical Guarantees

A Proof of Theorem 3.1

Here we will derive the upper bound of ERC of two-layer GATs with on-dimensional output for the single-time-step prediction.

As mentioned in section 2.2, our GATs model's second layer $l_{(2)}$ uses $\mathbf{Z}_{(1)} \in \mathbb{R}^{N\times KD_1}$ as input, and outputs $\mathbf{Z}_{(2)} \in \mathbb{R}^{N\times C}$. The input to the first layer $l_{(1)}$ is a set of node features

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \mathbb{R}^{N\times D},$$

where each $\mathbf{x}_i \in \mathbb{R}^D$, D is the number of features in each node. The first layer produces $\mathbf{Z}_{(1)} \in \mathbb{R}^{N\times KD_1}$ with each $\mathbf{z}^i_{(1)} \in \mathbb{R}^{KD_1}$ for $i = 1, \ldots, N$.

We have the first layer's weight matrix to be

$$\mathbf{W}_{(1,k)} = \left[\mathbf{w}^1_{(1,k)}, \ldots, \mathbf{w}^{D_1}_{(1,k)}\right] \in \mathbb{R}^{D\times D_1}.$$

Now we write the output vector of GATs from (2) for all node i, $i \in \mathcal{N}$ as,

$$\mathbf{z}^i_{(2)} = \sum_{kd_1=1}^{KD_1} \mathbf{w}^{kd_1}_{(2)} \sum_{i_2\in\mathcal{N}_i} p^{i,i_2}_{(2)} \cdot \sum_{k=1}^{K} \sigma\left(\sum_{d=1}^{D} w^{d,d_1}_{(1,k)} \sum_{i_1\in\mathcal{N}_{i_2}} p^{i_2,i_1}_{(1,k)} x^d_{i_1}\right) \in \mathbb{R}^C.$$

Here $w^{d,d_1}_{(1,k)} = \left[\mathbf{W}_{(1,k)}\right]_{dd_1}$, the d, d_1's entry of $\mathbf{W}_{(1,k)}$. Here we use index notation $kd_1 \in [KD_1]$ because we already have indexes $k \in [K]$ and $d_1 \in [D_1]$. The summation indices are within the sets $i_2 \in \mathcal{N}_i$ and $i_1 \in \mathcal{N}_{i_2}$. By the definition of the attention matrix $\mathbf{P}$ as given in (6), we can allow both indices i_2 and i_1 to iterate from 1 to N without losing anything.

And it is easy to show that above output can be easily written as in vector format:

$$\mathbf{z}^i_{(2)} = \sum_{kd_1=1}^{KD_1} \mathbf{w}^{kd_1}_{(2)} \sum_{i_2=1}^{N} p^{i,i_2}_{(2)} \cdot \sum_{k=1}^{K} \sigma\left(\sum_{i_1=1}^{N} p^{i_2,i_1}_{(1,k)} \left\langle \mathbf{w}^{d_1}_{(1,k)}, \mathbf{x}_{i_1}\right\rangle\right),$$

where $\mathbf{w}^{d_1}_{(1,k)}$ represents the column d_1 of $\mathbf{W}_{(1,k)}$, $\mathbf{w}^{kd_1}_{(2)}$ represents the row kd_1 of $\mathbf{W}_{(2)}$, and $\mathbf{w}^c_{(2)}$ represents the column c of $\mathbf{W}_{(2)}$. Then the class of functions defined over the node set $\mathcal{N}$ for all $i \in \mathcal{N}$ and $c \in [C]$ will be

$$\mathcal{F}_i = \Bigg\{ f : \mathbf{X} \mapsto \mathbf{e}_i^\top f(\mathbf{X}) = \sum_{kd_1=1}^{KD_1} \mathbf{w}^{kd_1}_{(2)} \sum_{i_2=1}^{N} p^{i,i_2}_{(2)} \cdot \sum_{k=1}^{K} \sigma\left(\sum_{i_1=1}^{N} p^{i_2,i_1}_{(1,k)} \left\langle \mathbf{w}^{d_1}_{(1,k)}, \mathbf{x}_{i_1}\right\rangle\right) \in \mathbb{R}^C; \tag{22}$$

$$\left\|\mathbf{W}_{(1,k)}\right\|_F \le M_1, \left\|\mathbf{w}^c_{(2)}\right\| \le M_2 \Bigg\}, \tag{23}$$

Furthermore, we also provide a model space with a single dimensional output that corresponds to the c-th component of model output from $f(\mathbf{x})$ for the c-th time step. Now we write the output of node i of time-step c of $l_{(2)}$ as

$$f^c_i(\mathbf{X}) = z^{i,c}_{(2)} = \sum_{kd_1=1}^{KD_1} w^{kd_1,c}_{(2)} \sum_{i_2=1}^{N} p^{i,i_2}_{(2)} \cdot \sum_{k=1}^{K} \sigma\left(\sum_{d=1}^{D} w^{d,d_1}_{(1,k)} \sum_{i_1=1}^{N} p^{i_2,i_1}_{(1,k)} x^d_{i_1}\right),$$

and its vector format is

$$\begin{aligned} f^c_i(\mathbf{X}) = z^{i,c}_{(2)} &= \sum_{kd_1=1}^{KD_1} w^{kd_1,c}_{(2)} \sum_{i_2=1}^{N} p^{i,i_2}_{(2)} \cdot \sum_{k=1}^{K} \sigma\left(\sum_{i_1=1}^{N} p^{i_2,i_1}_{(1,k)} \left\langle \mathbf{w}^{d_1}_{(1,k)}, \mathbf{x}_{i_1}\right\rangle\right) \\ &= \sum_{i_2=1}^{N} p^{i,i_2}_{(2)} \cdot \Big\langle \underbrace{\sum_{k=1}^{K} \sigma\left(\sum_{i_1=1}^{N} p^{i_2,i_1}_{(1,k)} \left\langle \mathbf{w}^{d_1}_{(1,k)}, \mathbf{x}_{i_1}\right\rangle\right)}_{:=\mathbf{z}^{i_2}_{(1)}}, \mathbf{w}^c_{(2)} \Big\rangle. \end{aligned} \tag{24}$$

Then the $c-th$ component of the output vector of node i can be re-written as

$$z_{(2)}^{i,c} = \sigma\left(\sum_{i_2=1}^{N} p_{(2)}^{i,i_2} \cdot \left\langle \mathbf{z}_{(1)}^{i_2}, \mathbf{w}_{(2)}^{c} \right\rangle\right),$$

where $\mathbf{w}_{(2)}^{c}$ represents column c of $\mathbf{W}_{(2)}$.

Now let hypothesis class $\mathcal{F}_i^c$ be a set of functions on $\mathbf{e}_i$ and $\mathbf{X}$. Specifically, we have such single dimensional GAT function space defined as

$$\begin{aligned}\mathcal{F}_i^c = \Bigg\{ f_i^c : \mathbf{X}_j \mapsto f_i^c(\mathbf{X}_j) = \sum_{kd_1=1}^{KD_1} w_{(2)}^{kd_1,c} \sum_{i_2=1}^{N} p_{(2)}^{i,i_2} \cdot \sum_{k=1}^{K} \sigma\left(\sum_{i_1=1}^{N} p_{(1,k)}^{i_2,i_1} \left\langle \mathbf{w}_{(1,k)}^{d_1}, \mathbf{x}_{i_1} \right\rangle\right); \\ \left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1, \left\|\mathbf{w}_{(2)}^{c}\right\| \leq M_2, f_i = (f_1^c, \ldots, f_i^c) \in \mathcal{F}_i \Bigg\}.\end{aligned} \tag{25}$$

We now begin presenting the proofs.

Step1: According to the definition of $\mathcal{F}_i^c$ in definition 25, the two layers GATs output in (24), and ERC in 2.4, fix a $\lambda > 0$, and a fix a node i for all $i = 1, \ldots, N$, and fix a c for all $c = 1, \ldots, C$, we have

$$\begin{aligned}\mathcal{R}(\mathcal{F}_i^c) =& \mathbb{E}_\epsilon \left[\frac{1}{n} \sup_{f_i^c \in \mathcal{F}_i^c} \sum_{j=1}^{n} \epsilon_j f_i^c(\mathbf{X}_j)\right] \\ =& \mathbb{E}_\epsilon \left[\frac{1}{n} \sup_{\substack{\left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1 \\ \left\|\mathbf{w}_{(2)}^{c}\right\| \leq M_2}} \sum_{j=1}^{n} \epsilon_j \sum_{i_2=1}^{N} p_{(2)j}^{i,i_2} \cdot \left\langle \mathbf{z}_{(1)j}^{i_2}, \mathbf{w}_{(2)}^{c} \right\rangle\right] \\ =& \mathbb{E}_\epsilon \left[\frac{1}{n} \sup_{\substack{\left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1 \\ \left\|\mathbf{w}_{(2)}^{c}\right\| \leq M_2}} \left\langle \sum_{j=1}^{n} \epsilon_j \sum_{i_2=1}^{N} p_{(2)j}^{i,i_2} \cdot \mathbf{z}_{(1)j}^{i_2}, \mathbf{w}_{(2)}^{c} \right\rangle\right] \\ \overset{(a)}{\leq}& \frac{1}{n}\frac{1}{\lambda} \log \mathbb{E}_\epsilon \left[\exp\left(\sup_{\substack{\left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1 \\ \left\|\mathbf{w}_{(2)}^{c}\right\| \leq M_2}} \lambda \left\langle \sum_{j=1}^{n} \epsilon_j \sum_{i_2=1}^{N} p_{(2)j}^{i,i_2} \cdot \mathbf{z}_{(1)j}^{i_2}, \mathbf{w}_{(2)}^{c} \right\rangle\right)\right]\end{aligned} \tag{26}$$

Inequality (a) is based on Jensen's inequality.

Equation (26) will be used in the derivation of the upper bound for the Rademacher complexity of GAT.

We continue to analyze the

$$Q \coloneqq \frac{1}{\lambda} \log \mathbb{E}_\epsilon \left[\exp\left(\sup_{\substack{\left\|\mathbf{W}_{(1,k)}\right\|_F \leq M_1 \\ \left\|\mathbf{w}_{(2)}^{c}\right\| \leq M_2}} \lambda \left\langle \sum_{j=1}^{n} \epsilon_j \sum_{i_2=1}^{N} p_{(2)j}^{i,i_2} \cdot \mathbf{z}_{(1)j}^{i_2}, \mathbf{w}_{(2)}^{c} \right\rangle\right)\right] \tag{27}$$

of the above equation.

By the argument in $\sup(\sum_i x_i) \leq \sum_i \sup(x_i)$, we have

$$\begin{aligned}
&\mathbb{E}_\epsilon \left[\exp \left(\sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \lambda \Big\langle \sum_{j=1}^n \epsilon_j \sum_{i_2=1}^N p^{i,i_2}_{(2)j} \cdot \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \Big\rangle \right) \right] \\
=&\mathbb{E}_\epsilon \left[\exp \left(\sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \lambda \sum_{i_2=1}^N \sum_{j=1}^n \epsilon_j p^{i,i_2}_{(2)j} \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right] \\
\leq&\mathbb{E}_\epsilon \left[\exp \left(\sum_{i_2=1}^N \sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \lambda \sum_{j=1}^n \epsilon_j p^{i,i_2}_{(2)j} \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right] \\
\leq&\frac{1}{N} \sum_{i_2=1}^N \mathbb{E}_\epsilon \left[\exp \left(\sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \lambda N \sum_{j=1}^n \epsilon_j p^{i,i_2}_{(2)j} \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right]
\end{aligned} \tag{28}$$

For any fixed i_2 for all j, we have

$$\begin{aligned}
&\mathbb{E}_\epsilon \left[\exp \left(\sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \lambda N \sum_{j=1}^n \epsilon_j p^{i,i_2}_{(2)j} \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right] \\
\overset{\text{(a)}}{\leq}&\mathbb{E}_\epsilon \left[\exp \left(\max_{j,i,i_2} \left| p^{i,i_2}_{(2)j} \right| \cdot \sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \lambda N \sum_{j=1}^n \epsilon_j \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right] \\
\overset{\text{(b)}}{\leq}&\mathbb{E}_\epsilon \left[\sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \exp \left(\lambda N \sum_{j=1}^n \epsilon_j \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right]
\end{aligned} \tag{29}$$

The inequality (a) is due to the contraction inequality of ERC which is presented in Lemma D.1. The inequality (b) is due to: the definition of graph attention matrix $\mathbf{P}$, i.e., the maximum value of entries in each row is equal to 1; the exponential function is monotone increasing.

Taking one step further, we obtain that

$$\begin{aligned}
&\mathbb{E}_\epsilon \left[\sup_{\substack{\|\mathbf{W}_{(1,k)}\|_F \leq M_1 \\ \|\mathbf{w}^c_{(2)}\| \leq M_2}} \exp \left(\lambda N \sum_{j=1}^n \epsilon_j \cdot \left\langle \mathbf{z}^{i_2}_{(1)j}, \mathbf{w}^c_{(2)} \right\rangle \right) \right] \\
\overset{\text{(a)}}{\leq}&\mathbb{E}_\epsilon \left[\sup_{\|\mathbf{W}_{(1,k)}\|_F \leq M_1} \exp \left(M_2 \lambda N \left\| \sum_{j=1}^n \epsilon_j \mathbf{z}^{i_2}_{(1)j} \right\| \right) \right]
\end{aligned} \tag{30}$$

The inequality (a) comes from the Cauchy-Schwartz inequality.

Step2: To bound the rest, we show a useful lemma.

Lemma A.1. *Given a hypothesis class $\mathcal{F}$ of vector-valued functions that map the GATs inputs to the outputs , and any convex and monotonically increasing function $h : \mathbb{R} \to \mathbb{R}^+$, the following equation holds for the GATs model of definition 15 with a 1-Lipschitz, positive-homogeneous activation function $\sigma(\cdot)$,*

$$\begin{aligned}
&\mathbb{E}_\epsilon \left[\sup_{\|\mathbf{W}_{(1,k)}\|_F \le M_1} h\left(\left\| \sum_{j=1}^{n} \epsilon_j \mathbf{z}_{(1)j}^{i_2} \right\| \right) \right] \\
=&\mathbb{E}_\epsilon \left[\sup_{\|\mathbf{w}_{(1,k)}\| = M_1} h\left(\left\| \sum_{j=1}^{n} \epsilon_j \sigma\left(\sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1} \left\langle \mathbf{w}_{(1,k)}, \mathbf{x}_{i_1 j} \right\rangle \right) \right\| \right) \right] \\
\le&2\mathbb{E}_\epsilon \left[\sup_{\|\mathbf{w}_{(1,k)}\| = M_1} h\left(\sum_{k=1}^{K} \sum_{j=1}^{n} \epsilon_j \sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1} \left\langle \mathbf{w}_{(1,k)}, \mathbf{x}_{i_1 j} \right\rangle \right) \right],
\end{aligned} \tag{31}$$

where ϵ_j are independent Rademacher variables.

Proof. We first introduce some definitions.

The $\mathbf{Z}_{(1)}$ is composed by concatenation of K outputs from the K identically structured attention layers $l_{(1,k)}$, with each output denoted as

$$\mathbf{H}_{(1,k)} = \sigma(\mathbf{P}_{(1,k)} \mathbf{X} \mathbf{W}_{(1,k)}) \in \mathbb{R}^{N \times D_1}.$$

We then further write the output of attention head k of first layer $l_{(1)}$ as

$$\mathbf{H}_{(1,k)} = \left[\mathbf{h}_{(1,k,1)}, \mathbf{h}_{(1,k,2)}, \ldots, \mathbf{h}_{(1,k,N)} \right]^\top \in \mathbb{R}^{N \times D_1}.$$

The output of first layer $l_{(1)}$ is

$$\mathbf{Z}_{(1)} = \left[\mathbf{h}_{(1,k)}^1, \mathbf{h}_{(1,2)}^2, \ldots, \mathbf{h}_{(1,KD_1)}^N \right]^\top \in \mathbb{R}^{N \times KD_1}.$$

By the concatenation relationship, we have the row i_2 of $\mathbf{Z}_{(1)}$ to be

$$\mathbf{z}_{(1)}^{i_2} = \left[\mathbf{h}_{(1,1,i_2)}^\top, \mathbf{h}_{(1,2,i_2)}^\top, \ldots, \mathbf{h}_{(1,k,i_2)}^\top \right] \in \mathbb{R}^{KD_1}. \tag{32}$$

And we define each $\mathbf{h}_{(1,k,i_2)}$ as

$$\mathbf{h}_{(1,k,i_2)} = \left[h_{(1,k,i_2)}^1, h_{(1,k,i_2)}^2, \ldots, h_{(1,k,i_2)}^{D_1} \right]^\top \in \mathbb{R}^{D_1},$$

with each $h_{(1,k,i_2)}^{d_1} \in \mathbb{R}$ defined as

$$h_{(1,k,i_2)}^{d_1} = \sigma\left(\sum_{i_1=1}^{N} p_{(1,k)}^{i_2,i_1} \left\langle \mathbf{w}_{(1,k)}^{d_1}, \mathbf{x}_{i_1} \right\rangle \right).$$

Firstly, we have

$$\begin{aligned}
&\mathbb{E}_\epsilon \left[\sup_{\|\mathbf{W}_{(1,k)}\|_F \le M_1} h\left(\left\| \sum_{j=1}^{n} \epsilon_j \cdot \mathbf{z}_{(1)j}^{i_2} \right\| \right) \right] \\
\le&\mathbb{E}_\epsilon \left[\sup_{\|\mathbf{w}_{(1,k)}\| = M_1} h\left(\sum_{k=1}^{K} \left| \sum_{j=1}^{n} \epsilon_j \sigma\left(\sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1} \left\langle \mathbf{w}_{(1,k)}, \mathbf{x}_{i_1 j} \right\rangle \right) \right| \right) \right],
\end{aligned} \tag{33}$$

The equality is due to the following derivation:

$$\left\|\sum_{j=1}^{n}\epsilon_j\cdot\mathbf{z}_{(1)j}^{i_2}\right\|^2=\sum_{kd_1=1}^{KD_1}\left(\sum_{j=1}^{n}\epsilon_j h_{(1,k,i_2)j}^{d_1}\right)^2=\sum_{k=1}^{K}\sum_{d_1=1}^{D_1}\left(\sum_{j=1}^{n}\epsilon_j h_{(1,k,i_2)j}^{d_1}\right)^2$$
$$=\sum_{k=1}^{K}\sum_{d_1=1}^{D_1}\left(\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)}^{d_1},\mathbf{x}_{i_1j}\right\rangle\right)\right)^2$$

For a fixed k-th attention head, we let the $\mathbf{w}_{(1,k)}^1,\mathbf{w}_{(1,k)}^2,\ldots,\mathbf{w}_{(1,k)}^{D_1}$ be the the columns of $\mathbf{W}_{(1,k)}$, then, by positive homogeneity of σ, we have

$$\sum_{k=1}^{K}\sum_{d_1=1}^{D_1}\left(\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)}^{d_1},\mathbf{x}_{i_1j}\right\rangle\right)\right)^2$$
$$=\sum_{k=1}^{K}\sum_{d_1=1}^{D_1}\left\|\mathbf{w}_{(1,k)}^{d_1}\right\|^2\left(\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\frac{\mathbf{w}_{(1,k)}^{d_1}}{\left\|\mathbf{w}_{(1,k)}^{d_1}\right\|},\mathbf{x}_{i_1j}\right\rangle\right)\right)^2$$

The supremum of this quantity over $\mathbf{w}_{(1,k)}^1,\mathbf{w}_{(1,k)}^2,\ldots,\mathbf{w}_{(1,k)}^{D_1}$ under the constraint that $\left\|\mathbf{W}_{(1,k)}\right\|_F^2\le M_1^2=\sum_{d_1=1}^{D_1}\left\|\mathbf{w}_{(1,k)}^{d_1}\right\|^2\le M_1^2$ is attained when $\left\|\mathbf{w}_{(1,k)}^{d_1}\right\|=M_1$ for some d_1 and $\left\|\mathbf{w}_{(1,k)}^{d_1'}\right\|=0$ for all other $d_1'\neq d_1$. In the end, only the d_1 term remain. For simplicity of notation, we use $\mathbf{w}_{(1,k)}$ to mean that d_1's column $\mathbf{w}_{(1,k)}^{d_1}$. Therefore, take the squared root on both sides, we have

$$\begin{aligned}&\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{W}_{(1,k)}\right\|_F\le M_1}h\left(\left\|\sum_{j=1}^{n}\epsilon_j\cdot\mathbf{z}_{(1)j}^{i_2}\right\|\right)\right]\\ \le&\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}h\left(\left|\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)},\mathbf{x}_{i_1j}\right\rangle\right)\right|\right)\right]\end{aligned}\tag{34}$$

Based on the previous analysis and go back to function h, we have

$$\begin{aligned}&\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}h\left(\left|\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)},\mathbf{x}_{i_1j}\right\rangle\right)\right|\right)\right]\\ \overset{(a)}{\le}&\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}h\left(+\left(\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)},\mathbf{x}_{i_1j}\right\rangle\right)\right)\right)\right]\\ &+\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}h\left(-\left(\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)},\mathbf{x}_{i_1j}\right\rangle\right)\right)\right)\right]\\ \overset{(b)}{=}&2\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}h\left(\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sigma\left(\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)},\mathbf{x}_{i_1j}\right\rangle\right)\right)\right]\\ \overset{(c)}{\le}&2\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}h\left(\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sum_{i_1=1}^{N}p_{(1,k)j}^{i_2,i_1}\left\langle\mathbf{w}_{(1,k)},\mathbf{x}_{i_1j}\right\rangle\right)\right]\end{aligned}\tag{35}$$

where the equality (a) above is due to $h(|x|) \leq h(+x) + h(-x)$ for any convex and monotonically increasing function h, and the equality (b) comes from the symmetry in the distribution of the ϵ_i random variables, and the equality (c) we use the fact that σ is $1-$Lipschitz continuous and Ledoux-Talagrand contraction inequality in Lemma D.1. □

Then by Lemma A.1, we choose $h(\alpha) = \exp(M_2\lambda N \cdot \alpha)$, we have

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|_F \leq M_1} \exp\left(M_2\lambda N \left\|\sum_{j=1}^{n}\epsilon_j \mathbf{z}_{(1)j}^{i_2}\right\|\right)\right] \\
\leq & 2\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda N \cdot \sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j \sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1}\left\langle \mathbf{W}_{(1,k)}, \mathbf{x}_{i_1 j}\right\rangle\right)\right] \\
\leq & 2\frac{1}{K}\sum_{k=1}^{K}\cdot\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda NK \cdot \sum_{j=1}^{n}\epsilon_j \sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1}\left\langle \mathbf{W}_{(1,k)}, \mathbf{x}_{i_1 j}\right\rangle\right)\right]
\end{aligned} \tag{36}$$

Put (28), (30), and (36) together, we get

$$\begin{aligned}
Q \leq & \frac{1}{\lambda}\log\left(\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|_F \leq M_1} \exp\left(M_2\lambda NK \left\|\sum_{j=1}^{n}\epsilon_j \cdot \mathbf{z}_{(1)j}^{i_2}\right\|_2\right)\right]\right) \\
\leq & \frac{1}{\lambda}\log\left(2\frac{1}{N}\sum_{i_2=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\cdot\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda NK \cdot \sum_{j=1}^{n}\epsilon_j \sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1}\left\langle \mathbf{W}_{(1,k)}, \mathbf{x}_{i_1 j}\right\rangle\right)\right]\right)
\end{aligned} \tag{37}$$

We continue to derive that

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda NK \cdot \sum_{j=1}^{n}\epsilon_j \sum_{i_1=1}^{N} p_{(1,k)j}^{i_2,i_1}\left\langle \mathbf{W}_{(1,k)}, \mathbf{x}_{i_1 j}\right\rangle\right)\right] \\
\leq & \frac{1}{N}\sum_{i_1=1}^{N}\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda NKN\left\langle \sum_{j=1}^{n}\epsilon_j p_{(1,k)j}^{i_2,i_1}\mathbf{x}_{i_1 j}, \mathbf{W}_{(1,k)}\right\rangle\right)\right]
\end{aligned} \tag{38}$$

Now we fix a i, we have

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda NKN\left\langle \sum_{j=1}^{n}\epsilon_j p_{(1,k)j}^{i_2,i_1}\mathbf{x}_{i_1 j}, \mathbf{W}_{(1,k)}\right\rangle\right)\right] \\
\leq & \mathbb{E}_\epsilon\left[\exp\left(\max_{k,j,i_2,i_1}\left|p_{(1,k)j}^{i_2,i_1}\right| \sup_{\|\mathbf{W}_{(1,k)}\|=M_1} M_2\lambda NKN\left\langle \sum_{j=1}^{n}\epsilon_j \mathbf{x}_{i_1 j}, \mathbf{W}_{(1,k)}\right\rangle\right)\right] \\
\overset{(a)}{\leq} & \mathbb{E}_\epsilon\left[\sup_{\|\mathbf{W}_{(1,k)}\|=M_1} \exp\left(M_2\lambda NKN\left\langle \sum_{j=1}^{n}\epsilon_j \mathbf{x}_{i_1 j}, \mathbf{W}_{(1,k)}\right\rangle\right)\right],
\end{aligned} \tag{39}$$

where in inequality a again we use the property of the attention matrix $\mathbf{P}$.

Step3: Then by Cauchy-Schwartz, we have

$$
\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}\exp\left(M_2\lambda NKN\Big\langle\sum_{j=1}^{n}\epsilon_j\mathbf{x}_{i_1j},\mathbf{w}_{(1,k)}\Big\rangle\right)\right]\\
=&\mathbb{E}_\epsilon\left[\sup_{\left\|\mathbf{w}_{(1,k)}\right\|=M_1}\exp\left(M_2\lambda NKN\left(\left\|\sum_{j=1}^{n}\epsilon_j\mathbf{x}_{i_1j}\right\|_2^2\left\|\mathbf{w}_{(1,k)}\right\|_2^2\right)^{1/2}\right)\right]\\
\overset{(a)}{\leq}&\mathbb{E}_\epsilon\left[\exp\left(M_2M_1\lambda NKN\left\|\sum_{j=1}^{n}\epsilon_j\mathbf{x}_{i_1j}\right\|\right)\right].
\end{aligned}
\tag{40}
$$

here we use $\mathbf{x}_{i_1j}$ to mean the input coming from the j's batch of data and n's node. Subsequently, we define the following random variable Z

$$
Z = M_2M_1N^2K\left\|\sum_{j=1}^{n}\epsilon_j\mathbf{x}_{i_1j}\right\| \tag{41}
$$

We further bound the expectation of Z, we have

$$
\begin{aligned}
\mathbb{E}_\epsilon[Z] &= \mathbb{E}_\epsilon\left[M_2M_1N^2K\left\|\sum_{j=1}^{n}\epsilon_j\mathbf{x}_{i_1j}\right\|\right]\\
&\overset{(a)}{\leq} M_2M_1N^2K\sqrt{\mathbb{E}_\epsilon\left[\left\|\sum_{j=1}^{n}\epsilon_j\mathbf{x}_{i_1j}\right\|^2\right]}\\
&\leq M_2M_1N^2K\sqrt{\mathbb{E}_\epsilon\left[\sum_{j'j}\epsilon_{j'}\epsilon_j\mathbf{x}_{i_1j'}^\top\mathbf{x}_{i_1j}\right]}\\
&\overset{(b)}{=} M_2M_1N^2K\sqrt{\sum_{j=1}^{n}\left\|\mathbf{x}_{i_1j}\right\|^2}
\end{aligned}
\tag{42}
$$

and the inequality (a) is by Jensen's inequality. And the equality (b) follows the i.i.d. condition of Rademacher sequences with zero-mean.

Next, based on equation (37), (39), (40), and (41),

We have

$$
\begin{aligned}
n\mathcal{R}(\mathcal{F}_i^c) \leq Q &\leq \frac{1}{\lambda}\log\left(2\frac{1}{N}\sum_{i_2=1}^{N}\frac{1}{K}\sum_{k=1}^{K}\frac{1}{N}\sum_{i_1=1}^{N}\cdot\mathbb{E}_\epsilon\left[\exp\left(\lambda Z\right)\right]\right)\\
&= \frac{1}{\lambda}\log 2 + \frac{1}{\lambda}\log\left(\cdot\mathbb{E}_\epsilon\left[\exp\lambda\left(Z-\mathbb{E}_\epsilon\left[Z\right]\right)\right]\right) + \mathbb{E}_\epsilon\left[Z\right]
\end{aligned}
\tag{43}
$$

Following the results in Ledoux & Talagrand (1991), we can show Z is sub-Gaussian with the following variance factor V since Z is a deterministic function of the i.i.d. random variables $\epsilon_1,\ldots,\epsilon_n$ satisfies a bounded-difference condition, i.e.,

$$
Z(\epsilon_1,...,\epsilon_j,...,\epsilon_n) - Z(\epsilon_1,...,-\epsilon_j,...,\epsilon_n) \leq 2M_2M_1N^2K\left\|\mathbf{x}_{i_1j}\right\| \tag{44}
$$

If we denote $M = M_2M_1N^2K$, then we have

$$
V = \frac{1}{4}\sum_{j=1}^{n}\left(2M\left\|\mathbf{x}\right\|\right)^2 = 2M^2\sum_{j=1}^{n}\left\|\mathbf{x}_{i_1j}\right\|^2 \tag{45}
$$

According to the property of sub-Gaussian random variables, the following inequality holds for Z:

$$\begin{aligned}&\frac{1}{\lambda}\log\left(\mathbb{E}_\epsilon\left[\exp\left(\lambda(Z-\mathbb{E}_\epsilon\left[Z\right])\right)\right]\right)\\ \leq&\frac{\lambda M^2\sum_{j=1}^n\left\|\mathbf{x}_{i_1j}\right\|^2}{2}\end{aligned}\tag{46}$$

To minimize the right hand side of (43), let $\lambda=\frac{\sqrt{2\log(2)}}{M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1j}\right\|^2}}$, combined with the earlier result, we get

$$\begin{aligned}Q\overset{(a)}{\leq}&\frac{1}{\lambda}\log 2+\frac{1}{\lambda}\log\left(\cdot\mathbb{E}_\epsilon\left[\exp\lambda\left(Z-\mathbb{E}_\epsilon\left[Z\right]\right)\right]\right)+\mathbb{E}_\epsilon\left[Z\right]\\ \leq&M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1j}\right\|^2}\sqrt{\frac{\log 2}{2}}+\sqrt{\frac{\log 2}{2}}M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1j}\right\|^2}+M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1j}\right\|^2}\\ \leq&M\left(\sqrt{2\log 2}+1\right)\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1j}\right\|^2}\\ \overset{(b)}{\leq}&N^2KM_2M_1\sqrt{n}\left(\sqrt{2\log 2}+1\right)B\end{aligned}\tag{47}$$

where (a) holds by the choice of λ and simple calculation, and (b) holds by our assumption that $\|\mathbf{x}\|\leq B$. Finally, after changing N to N_e, and (47) holds for all n, we have

$$\mathcal{R}(\mathcal{F}_i^c)\leq BK(N_e)^2M_1M_2\left(\sqrt{2\log 2}+1\right)(n)^{-1/2}\tag{48}$$

B Poof of Theorem 3.2

We now turn to prove the Theorem 3.2. Our proof strategy will be the following. We first provide a classic theorem that was used to bound the expected loss based on the empirical loss and the upper bound of ERC of loss functions associated with GATs for the multi-time-step situation $\mathcal{F}_i$, then we derive this upper bound of ERC of loss functions of $\mathcal{F}_i$ by extending the upper bound of ERC of GATs with one-dimensional output for the single-time-step prediction $\mathcal{F}_i^c$, based on the existing theorem in literature.

Proof. From Theorem 3.3 Mohri et al. (2018), first, we define loss function class for $\mathcal{F}_i$

$$\begin{aligned}\mathcal{G}_{\mathcal{F}_i}=\{&(\mathbf{X},\mathbf{y})\mapsto g(f_i(\mathbf{X}),\mathbf{y});\\ &f_i\in\mathcal{F}_i,\mathbf{X}\in\mathbb{R}^{N\times D},\mathbf{y}\in\mathbb{R}^C\}.\end{aligned}\tag{49}$$

We will show the following holds for all $g\in\mathcal{G}_{\mathcal{F}}$,

$$\mathbf{E}(g)\leq\hat{\mathbf{E}}(g)+2\frac{1}{N}\sum_{i=1}^N\mathcal{R}(\mathcal{G}_{\mathcal{F}_i})+3\sqrt{\frac{\ln(2/\delta)}{2n}},$$

where $\mathbf{E}(g)$, $\mathcal{R}(\mathcal{G}_{\mathcal{F}_i})$, and $\hat{\mathbf{E}}(g)$ are defined in section 2.3. We will prove this based on Theorem 3.3.

We know the training set contains training set $S=\{(\mathbf{X}_1,\mathbf{Y}_1),\ldots,(\mathbf{X}_n,\mathbf{Y}_n)\}$ contains n size of i.i.d samples over the graph $\mathcal{G}$, where for each sample $j=1,\ldots,n$, we have feature $\mathbf{X}_j=[\mathbf{x}_{1j},\mathbf{x}_{2j},\ldots,\mathbf{x}_{Nj}]^\top$, true label $\mathbf{Y}_j=[\mathbf{y}_{1j},\mathbf{y}_{2j},\ldots,\mathbf{y}_{Nj}]^\top$. For $\mathcal{G}_{\mathcal{F}}$, we let ϕ defined for any sample S by

$$\phi(S)=\sup_{g\in\mathcal{G}_{\mathcal{F}}}\left(\mathbf{E}(g)-\hat{\mathbf{E}}_{nS}(g)\right).\tag{50}$$

Let S and S' be two samples differing by exactly one point, say $(\mathbf{X}_j,\mathbf{Y}_j)$ in S and $(\mathbf{X}'_j,\mathbf{Y}'_j)$ in S'. Furthermore, we have $S=\{(\mathbf{X}_1,\mathbf{Y}_1),\ldots,(\mathbf{X}_n,\mathbf{Y}_n)\}$ and $S'=\{(\mathbf{X}'_1,\mathbf{Y}'_1),\ldots,(\mathbf{X}'_n,\mathbf{Y}'_n)\}$ having the same distribution as $(\mathbf{X},\mathbf{Y})$.

Then, since the difference of suprema does not exceed the supremum of the difference, we have

$$\phi(S') - \phi(S) \leq \sup_{g \in \mathcal{G}_{\mathcal{F}}} \left(\hat{\mathbf{E}}_{nS'}(g) - \hat{\mathbf{E}}_{nS}(g) \right) = \sup_{g \in \mathcal{G}_{\mathcal{F}}} \frac{\frac{1}{N} \sum_{i=1}^{N} \left[g(\mathbf{e}_i^\top f(\mathbf{X}'_j), \mathbf{y}'_{ij})) - g(\mathbf{e}_i^\top f(\mathbf{X}_j), \mathbf{y}_{ij})) \right]}{n} \leq \frac{1}{n}. \quad (51)$$

Similarly, we can obtain $\phi(S) - \phi(S') \leq \frac{1}{n}$, thus $|\phi(S) - \phi(S')| \leq \frac{1}{n}$. Then, by Bounded differences inequality, see Corollary 2.21 in Wainwright (2019), for any $\delta > 0$, with probability at least $1 - \delta/2$, the following holds:

$$\phi(S) \leq \mathbb{E}_S \left[\phi(S) \right] + \sqrt{\frac{\log \frac{2}{\delta}}{2n}}. \quad (52)$$

We next bound the expectation of the right-hand side as follows:

$$\begin{aligned}
\mathbb{E}_S \left[\phi(S) \right] &= \mathbb{E} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}}} \left(\mathbf{E}(g) - \hat{\mathbf{E}}_{nS}(g) \right) \right] \\
&= \mathbb{E}_S \left[\sup_{g \in \mathcal{G}_{\mathcal{F}}} \left(\mathbb{E} \left[\frac{1}{N} \sum_{i=1}^{N} g \left(\mathbf{e}_i^\top f(\mathbf{X}), \mathbf{y}_i \right) \right] - \frac{1}{n} \sum_{j=1}^{n} \frac{1}{N} \sum_{i=1}^{N} g \left(\mathbf{e}_i^\top f(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right) \right] \\
&\leq \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_S \left[\sup_{g \in \mathcal{G}_{\mathcal{F}}} \left(\mathbb{E} \left[g \left(\mathbf{e}_i^\top f(\mathbf{X}), \mathbf{y}_i \right) \right] - \frac{1}{n} \sum_{j=1}^{n} g \left(\mathbf{e}_i^\top f(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right) \right] \\
&\overset{(a)}{=} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_S \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \left(\mathbb{E}_{S'} \left[\frac{1}{n} \sum_{j=1}^{n} g \left(f_i(\mathbf{X}'_j), \mathbf{y}'_{ij} \right) - \frac{1}{n} \sum_{j=1}^{n} g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right] \right) \right] \\
&\overset{(b)}{\leq} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{S,S'} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \left(\frac{1}{n} \sum_{j=1}^{n} \left(g \left(f_i(\mathbf{X}'_j), \mathbf{y}'_{ij} \right) - g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right) \right) \right] \\
&\overset{(c)}{\leq} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{S,S',\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \left(\frac{1}{n} \sum_{j=1}^{n} \epsilon_j \cdot \left(g \left(f_i(\mathbf{X}'_j), \mathbf{y}'_{ij} \right) - g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right) \right) \right] \\
&\leq \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{S,\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g \left(f_i(\mathbf{X}'_j), \mathbf{y}'_{ij} \right) \right] + \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{S',\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} -\epsilon_j g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right] \\
&= 2 \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{S,\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right] = 2 \frac{1}{N} \sum_{i=1}^{N} \mathbb{E} \, \mathcal{R}(\mathcal{G}_{\mathcal{F}_i}).
\end{aligned}$$

Equation (a) uses the fact that points in S' and S are sampled in an i.i.d. fashion and they have the same distribution. Inequality (b) holds due to the sub-additivity of the supremum function. Inequality (c) holds due to the $\left(g \left(f_i(\mathbf{X}'_j), \mathbf{y}'_{ij} \right) - g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right)$ and $\epsilon_j \cdot \left(g \left(f_i(\mathbf{X}'_j), \mathbf{y}'_{ij} \right) - g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right)$ have the same distribution.

Furthermore, we observe that, by changing one point in S changes $\mathbb{E}_{S,\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right]$ by at most $\frac{1}{n}$. Then, using again bounded difference inequality, with probability $1 - \delta/2$ the following holds:

$$\mathbb{E}_{S,\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right] \leq \mathbb{E}_{\epsilon} \left[\sup_{g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g \left(f_i(\mathbf{X}_j), \mathbf{y}_{ij} \right) \right] + \sqrt{\frac{\log \frac{2}{\delta}}{2n}}$$

Thus, we obtain with with probability $1 - \delta$ the following holds:

$$\sup_{g \in \mathcal{G}_{\mathcal{F}}} \left(\mathbf{E}(g) - \hat{\mathbf{E}}_{nS}(g) \right) \leq 2 \frac{1}{N} \sum_{i=1}^{N} \mathcal{R}(\mathcal{G}_{\mathcal{F}_i}) + 3 \sqrt{\frac{\log \frac{2}{\delta}}{2n}}.$$

In order to extend to multi-dimensional vector-valued functions for MTSF, we will use the contraction inequality for the hypothesis class $\mathcal{F}_i$ of vector-valued functions $f_i \in \mathbb{R}^C$ and Lipschitz continuous property of g. However, the RHS of equation 53 has supremum over all $f_i \in \mathcal{F}_i$ which is hard to compute and we can reduce it to scalar classes, and derive the following bound (Maurer, 2016). Lemma B.1 shows this derivation, Then put the result in (48) for $\mathcal{R}(\mathcal{F}_i^c)$ for all $i = 1, \ldots, N$ and for all $c = 1, \ldots, C$, in the end, we get

$$\sup_{g \in \mathcal{G}_\mathcal{F}} \left(\mathbf{E}(g) - \hat{\mathbf{E}}_{nS}(g) \right) \leq 2\sqrt{2} C L_g \mathcal{R}(\mathcal{F}_i^c) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2n}}.$$

□

Lemma B.1 (See Corollary 4 in Maurer (2016))**.** *Let $\mathcal{F}$ be the class of GATs functions in* (13)*. Let $\mathcal{F}_i$ be the class of vector-valued functions in* (14) *with $f_i = (f_i^1, \ldots, f_i^C) \in \mathcal{F}_i$, with each $f_i^c \in \mathcal{F}_i^c$ with $\mathcal{F}_i^c$ defined in* (15)*, let training set $S = \{(\mathbf{X}_1, \mathbf{Y}_1), \ldots, (\mathbf{X}_n, \mathbf{Y}_n)\}$ contains n size of samples, where for each sample $j \in 1, \ldots, n$, we have $\mathbf{X}_j = [\mathbf{x}_{1j}, \mathbf{x}_{2j}, \ldots, \mathbf{x}_{Nj}]^\top$, $\mathbf{Y}_j = [\mathbf{y}_{1j}, \mathbf{y}_{2j}, \ldots, \mathbf{y}_{Nj}]^\top$. with each $\mathbf{x}_{ij} \in \mathbb{R}^D$ and $\mathbf{y}_{ij} \in \mathbb{R}^C$. Let $g(\cdot, \cdot) \in \mathcal{G}_\mathcal{F}$ be a L_g-Lipschitz function mapping $\mathbb{R}^C \times \mathbb{R}^C$ to $[0, 1]$ with $\mathcal{G}_{\mathcal{F}_i}$ defined in* (49)*, and associated to $\mathcal{F}_i$. Then we have for all $i = 1, \ldots, N$,*

$$\mathcal{R}(\mathcal{G}_{\mathcal{F}_i}) = \mathbb{E}_\epsilon \left[\sup_{g \in g_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g(f_i(\mathbf{X}_j), \mathbf{y}_{ij}) \right] \leq \sqrt{2} \sum_{c=1}^{C} L_g \mathbb{E}_\epsilon \left[\sup_{f_i^c \in \mathcal{F}_i^c} \sum_{j=1}^{n} \epsilon_j f_i^c(\mathbf{X}_j) \right]. \tag{53}$$

Proof. Based on the fact that our loss g is bounded and continuous, it has Lipschitz norm 1. So we have

$$\begin{aligned} \mathcal{R}(\mathcal{G}_{\mathcal{F}_i}) &\coloneqq \mathbb{E}_\epsilon \left[\sup_{f_i \in \mathcal{F}_i, g \in \mathcal{G}_{\mathcal{F}_i}} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j g(f_i(\mathbf{X}_j), \mathbf{y}_{ij}) \right] \\ &= \mathbb{E}_\epsilon \left[\sup_{\substack{f_i \in \mathcal{F}_i \\ g \text{ has a } L_g\text{-Lipschitz norm}}} \sum_{j=1}^{n} \epsilon_j g(f_i(\mathbf{X}_j), \mathbf{y}_{ij}) \right] \\ &\leq \sqrt{2} L_g \mathbb{E}_\epsilon \left[\sup_{f_i \in \mathcal{F}_i} \frac{1}{n} \sum_{j=1}^{n} \sum_{c=1}^{C} \epsilon_{jc} f_i^c(\mathbf{X}_j) \right], \end{aligned} \tag{54}$$

where ϵ_{jc} is an independent doubly indexed Rademacher sequence.

$$\mathbb{E}_\epsilon \left[\sup_{\substack{f_i \in \mathcal{F}_i \\ f_i = (f_i^1, \ldots, f_i^c)}} \frac{1}{n} \sum_{j=1}^{n} \sum_{c=1}^{C} \epsilon_{jc} f_i^c(\mathbf{X}_j) \right] \leq \sum_{c=1}^{C} \mathbb{E}_\epsilon \left[\sup_{f_i^c \in \mathcal{F}_i^c} \frac{1}{n} \sum_{j=1}^{n} \epsilon_j f_i^c(\mathbf{X}_j) \right], \tag{55}$$

Then we can derive the following upper bound for the loss function $g(f_i(\mathbf{X}), \mathbf{y}_i)$ with $f_i(\mathbf{X})$ being vector-valued functions based on equation 54 and 55, where the RHS of equation 55 is related to $\mathcal{R}(\mathcal{F}_i^c)$:

$$\mathcal{R}(\mathcal{G}_{\mathcal{F}_i}) \leq \sqrt{2} \sum_{c=1}^{C} L_g \mathcal{R}(\mathcal{F}_i^c) \tag{56}$$

□

C Proof Theorem 4.1 and Theorem 4.2

Proof. For $l = L$, by the definition of network output and in (2) the ERC in (12), fix a $\lambda > 0$, and fix a node i for all $i = 1, \ldots, N$, and fix a c for all $c = 1, \ldots, C$, we have

$$\begin{aligned}
\mathcal{R}(\mathcal{F}_i^c) =& \mathbb{E}_\epsilon \left[\frac{1}{n} \sup_{f_i^c \in \mathcal{F}_i^c} \sum_{j=1}^{n} \epsilon_j f_i^c(\mathbf{X}_j) \right] \\
=& \mathbb{E}_\epsilon \left[\frac{1}{n} \sup_{\substack{f_{(L-1)} \in \mathcal{F}_{L-1} \\ \left\|\mathbf{w}_{(L)}^c\right\| \le M_L}} \sum_{j=1}^{n} \epsilon_j \sum_{i_L=1}^{N} p_{(L)j}^{i,i_L} \cdot \left\langle \mathbf{z}_{(L-1)j}^{i_L}, \mathbf{w}_{(L)}^c \right\rangle \right] \\
\le& \frac{1}{n} \frac{1}{\lambda} \log \mathbb{E}_\epsilon \left[\sup_{\substack{f_{(L-1)} \in \mathcal{F}_{L-1} \\ \left\|\mathbf{w}_{(L)}^c\right\| \le M_L}} \exp\left(\lambda \sum_{j=1}^{n} \epsilon_j \sum_{i_L=1}^{N} p_{(L)j}^{i,i_L} \cdot \left\langle \mathbf{z}_{(L-1)j}^{i_L}, \mathbf{w}_{(L)}^c \right\rangle \right) \right]
\end{aligned} \tag{57}$$

we continue to analyze the

$$Q \coloneqq \frac{1}{\lambda} \log \mathbb{E}_\epsilon \left[\exp\left(\sup_{\substack{\left\|\mathbf{W}_{(1,k)}\right\|_F \le M_1 \\ \left\|\mathbf{w}_{(2)}^c\right\| \le M_2}} \lambda \Big\langle \sum_{j=1}^{n} \epsilon_j \sum_{i_L=1}^{N} p_{(L)j}^{i,i_L} \cdot \mathbf{z}_{(L-1)j}^{i_L}, \mathbf{w}_{(L)}^c \Big\rangle \right) \right] \tag{58}$$

of the above equation.

We use the same analysis in Equation (28), we get

$$\begin{aligned}
& \mathbb{E}_\epsilon \left[\sup_{\substack{f_{(L-1)} \in \mathcal{F}_{L-1} \\ \left\|\mathbf{w}_{(L)}^c\right\| \le M_L}} \exp\left(\lambda \sum_{j=1}^{n} \epsilon_j \sum_{i_L=1}^{N} p_{(L)j}^{i,i_L} \cdot \left\langle \mathbf{z}_{(L-1)j}^{i_L}, \mathbf{w}_{(L)}^c \right\rangle \right) \right] \\
\le& \frac{1}{N} \sum_{i_L=1}^{N} \mathbb{E}_\epsilon \left[\sup_{\substack{\left\|\mathbf{W}_{(L-1,k)}\right\| \le M_{L-1} \\ f_{(L-2)} \in \mathcal{F}_{L-2}}} \exp\left(\lambda N \sum_{j=1}^{n} \epsilon_j \cdot \left\langle \mathbf{z}_{(L-1)j}^{i_L}, \mathbf{w}_{(L)}^c \right\rangle \right) \right].
\end{aligned}$$

We then use the same analysis in equation (29) and (30), for any fixed node index i_L, we get

$$\begin{aligned}
& \mathbb{E}_\epsilon \left[\sup_{\substack{\left\|\mathbf{W}_{(L-1,k)}\right\| \le M_{L-1} \\ f_{(L-2)} \in \mathcal{F}_{L-2}}} \exp\left(\lambda N \sum_{j=1}^{n} \epsilon_j \cdot \left\langle \mathbf{z}_{(L-1)j}^{i_L}, \mathbf{w}_{(L)}^c \right\rangle \right) \right] \\
\le& \mathbb{E}_\epsilon \left[\sup_{\substack{\left\|\mathbf{W}_{(L-1,k)}\right\| \le M_{L-1} \\ f_{(L-2)} \in \mathcal{F}_{L-2}}} \exp\left(M_L \lambda N \left\| \sum_{j=1}^{n} \epsilon_j \cdot \mathbf{z}_{(L-1)j}^{i_L} \right\| \right) \right].
\end{aligned} \tag{59}$$

By using the same analysis for equation (36), we get

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{W}_{(L-1,k)}\|\le M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda N\left\|\sum_{j=1}^{n}\epsilon_j\cdot\mathbf{z}_{(L-1)j}^{i_L}\right\|\right)\right]\\
\overset{(a)}{\le}&2\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{w}_{(L-1,k)}\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda N\sum_{k=1}^{K}\sum_{j=1}^{n}\epsilon_j\sum_{i_{L-1}=1}^{N}p_{(L-1,k)j}^{i_L,i_{L-1}}\left\langle\mathbf{w}_{(L-1,k)},\mathbf{z}_{(L-2)j}^{i_{L-1}}\right\rangle\right)\right]\\
\le&2\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{w}_{(L-1,k)}\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NK\sum_{j=1}^{n}\epsilon_j\sum_{i_{L-1}=1}^{N}p_{(L-1,k)j}^{i_L,i_{L-1}}\left\langle\mathbf{w}_{(L-1,k)},\mathbf{z}_{(L-2)j}^{i_{L-1}}\right\rangle\right)\right]
\end{aligned}\tag{60}$$

For the inequality (a) above, we again apply Lemma A.1.

Similar to the Inequality (38), for a fixed $k=1,\dots,K$, we have

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{w}_{(L-1,k)}\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NK\sum_{j=1}^{n}\epsilon_j\sum_{i_{L-1}=1}^{N}p_{(L-1,k)j}^{i_L,i_{L-1}}\left\langle\mathbf{w}_{(L-1,k)},\mathbf{z}_{(L-2)j}^{i_{L-1}}\right\rangle\right)\right]\\
\le&\frac{1}{N}\sum_{i_{L-1}=1}^{N}\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{w}_{(L-1,k)}\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NKN\sum_{j=1}^{n}\epsilon_j p_{(L-1,k)j}^{i_L,i_{L-1}}\left\langle\mathbf{w}_{(L-1,k)},\mathbf{z}_{(L-2)j}^{i_{L-1}}\right\rangle\right)\right]
\end{aligned}\tag{61}$$

For any fixed i_{L-1}, similar to the processes in equation (39), we have

$$\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{w}_{(L-1,k)}\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NKN\cdot\left\langle\sum_{j=1}^{n}\epsilon_j p_{(L-1,k)j}^{i_L,i_{L-1}}\mathbf{z}_{(L-2)j}^{i_{L-1}},\mathbf{w}_{(L-1,k)}\right\rangle\right)\right]\\
\overset{(a)}{\le}&\mathbb{E}_\epsilon\left[\sup_{\substack{\|\mathbf{w}_{(L-1,k)}\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NKN\left\langle\sum_{j=1}^{n}\epsilon_j\mathbf{z}_{(L-2)j}^{i_{L-1}},\mathbf{w}_{(L-1,k)}\right\rangle\right)\right],
\end{aligned}\tag{62}$$

where in inequality (a) again we use the property of the attention matrix $\mathbf{P}$.

Now, we have

$$
\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\substack{\left\|\mathbf{w}_{(L-1,k)}\right\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NKN\Big\langle\sum_{j=1}^n\epsilon_j\mathbf{z}_{(L-2)j}^{i_{L-1}},\mathbf{w}_{(L-1,k)}\Big\rangle\right)\right]\\
=&\mathbb{E}_\epsilon\left[\sup_{\substack{\left\|\mathbf{w}_{(L-1,k)}\right\|=M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda NKN\left(\left\|\sum_{j=1}^n\epsilon_j\mathbf{z}_{(L-2)j}^{i_{L-1}}\right\|_2^2\left\|\mathbf{w}_{(L-1,k)}\right\|_2^2\right)^{1/2}\right)\right]\\
\overset{(a)}{\leq}&\mathbb{E}_\epsilon\left[\sup_{\substack{\left\|\mathbf{W}_{(L-2,k)}\right\|_F\leq M_{L-2}\\ f_{(L-3)}\in\mathcal{F}_{L-3}}}\exp\left(M_LM_{L-1}\lambda NKN\left\|\sum_{j=1}^n\epsilon_j\mathbf{z}_{(L-2)j}^{i_{L-1}}\right\|\right)\right].
\end{aligned}
$$

Then we get the induction equation, for any $l\in[L-1]$, we have

$$
\begin{aligned}
&\mathbb{E}_\epsilon\left[\sup_{\substack{\left\|\mathbf{W}_{(L-1,k)}\right\|\leq M_{L-1}\\ f_{(L-2)}\in\mathcal{F}_{L-2}}}\exp\left(M_L\lambda N\left\|\sum_{j=1}^n\epsilon_j\cdot\mathbf{z}_{(L-1)j}^{i_L}\right\|\right)\right]\\
\leq&2^{L-l}\frac{1}{N^{L-l+1}}\frac{1}{K^{L-l}}\sum_{i_L=1}^N\cdots\sum_{i_l=1}^N\sum_{k_{L-1}=1}^K\cdots\sum_{k_l=1}^K\\
&\left\{\mathbb{E}_\epsilon\left[\sup_{\substack{\left\|\mathbf{W}_{(l-1,k)}\right\|_F\leq M_{l-1}\\ f_{(l-2)}\in\mathcal{F}_{l-2}}}\exp\left(M_L\dots M_lN^{L-l+1}K^{L-l}\lambda\left\|\sum_{j=1}^n\epsilon_j\mathbf{z}_{(l-1)j}^{i_{l-2}}\right\|\right)\right]\right\}
\end{aligned}
$$

By induction until $l=1$, we get

$$
\begin{aligned}
Q\leq\frac{1}{\lambda}\log\Bigg(&2^{L-1}\frac{1}{N^L}\frac{1}{K^{L-1}}\sum_{i_L=1}^N\cdots\sum_{i_1=1}^N\sum_{k_{L-1}=1}^K\cdots\sum_{k_1=1}^K\\
&\cdot\mathbb{E}_\epsilon\left[\exp\left(M_L\dots M_1N^LK^{L-1}\lambda\left\|\sum_{j=1}^n\epsilon_j\mathbf{x}_{i_1j}\right\|\right)\right]\Bigg).
\end{aligned}
\tag{63}
$$

We denote $M=M_L\dots M_1N^LK^{L-1}$, we define the random variable Z

$$
Z=M\left\|\sum_{j=1}^n\epsilon_j\mathbf{x}_{i_1j}\right\| \tag{64}
$$

Again, we can show that Z is sub-Gaussian satisfies a bounded-difference condition with the following variance factor V such that

$$
V=M^2\sum_{j=1}^n\|\mathbf{x}_{i_1j}\|^2 \tag{65}
$$

Combining with the results we derived in equation (57), (59), (63), and (64), we get

$$\begin{aligned} Q &\leq \frac{1}{\lambda}\log\left(2^{L-1}\cdot\mathbb{E}_\epsilon\left[\exp\left(\lambda Z\right)\right]\right) \\ &= \frac{(L-1)\log 2}{\lambda} + \frac{1}{\lambda}\left(\cdot\mathbb{E}_\epsilon\left[\exp\lambda\left(Z-\mathbb{E}_\epsilon\left[Z\right]\right)\right]\right) + \mathbb{E}_\epsilon\left[Z\right] \end{aligned} \tag{66}$$

Furthermore, by the property of sub-Gaussian, the following inequality holds for z:

$$\begin{aligned} &\frac{1}{\lambda}\log\left(\mathbb{E}_\epsilon\left[\exp\left(\lambda(Z-\mathbb{E}_\epsilon\left[Z\right])\right)\right]\right) \\ \leq &\frac{\lambda M^2\sum_{j=1}^n\left\|\mathbf{x}_{i_1 j}\right\|^2}{2} \end{aligned} \tag{67}$$

If we choose $\lambda = \frac{\sqrt{(L-1)2\log(2)}}{M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1 j}\right\|^2}}$, combined with the earlier result, we get

$$\begin{aligned} Q \leq &\frac{(L-1)\log 2}{\lambda} + \frac{1}{\lambda}\left(\cdot\mathbb{E}_\epsilon\left[\exp\lambda\left(Z-\mathbb{E}_\epsilon\left[Z\right]\right)\right]\right) + \mathbb{E}_\epsilon\left[Z\right] \\ \leq &M\sqrt{\sum_{j=1}^n\sum_{n=1}^N\left\|\mathbf{x}_{i_1 j}\right\|^2}\sqrt{(L-1)\log(2)/2} \\ &+\sqrt{(L-1)\log(2)/2}M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1 j}\right\|^2} + M\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1 j}\right\|^2} \\ \overset{(a)}{\leq} &R\left(\sqrt{2(L-1)\log(2)}+1\right)\sqrt{\sum_{j=1}^n\left\|\mathbf{x}_{i_1 j}\right\|^2} \\ \overset{(b)}{\leq} &N^L M_L\cdots M_1\sqrt{n}K^{L-1}\left(\sqrt{2(L-1)\log(2)}+1\right)B \end{aligned} \tag{68}$$

Finally, after changing N to N_e, and (68) holds for all n, we got

$$\mathcal{R}(\mathcal{F}^c) \leq N_e^L M_L\cdots M_1 K^{L-1}\left(\sqrt{2(L-1)\log(2)}+1\right)B(n)^{-1/2}. \tag{69}$$

□

D Auxiliary Lemmas

Lemma D.1. *(Ledoux-Talagrand contraction inequality) Let $h:\mathbb{R}\to[0,+\infty]$ be convex and monotonically increasing. Let $\phi_j:\mathbb{R}\to\mathbb{R}$ satisfy $\phi_j(0)=0$ and be Lipschitz with constant L, i.e., $|\phi_j(a)-\phi_j(b)|\leq L|a-b|$. Let ϵ_i be independent Rademacher random variables, and let f be a scalar function, then*

$$\mathbb{E}_\epsilon\left[h\left(\sup_{f\in\mathcal{F}}\sum_{j=1}^n\epsilon_j\phi_j(f(\mathbf{X}_j))\right)\right] \leq \mathbb{E}_\epsilon\left[h\left(L\sup_{f\in\mathcal{F}}\sum_{j=1}^n\epsilon_j f(\mathbf{X}_j)\right)\right] \tag{70}$$

Proof. See Theorem 7 in Duchi (2009). □

E Experiment and Data

E.1 Experiment setup

The model is trained by the Adam optimizer (Kingma & Ba, 2014). The learning rate is **1e-4**. The number of training epochs is **30**. The batch size is set to **5**. We split the dataset into three parts for training, validation and testing with a ratio of $0.6:0.2:0.2$. All the deep learning models, are implemented in Python with Pytorch and executed on a server with 8 NVIDIA GeForce GTX 2080Ti GPUs. The Nvidia driver version is 470.141.03 and the CUDA version is 11.4.

E.2 Experiment Data

The multivariate time series have about 1500 stocks, and all of these stocks are used for training and testing. This dataset is from Kaggle[2]. We ensure that the training and testing data do not overlap. For each stock, the one-week historical data is used to predict future returns. By default, we use a three-layer (input layer, hidden layer, and output layer) GAT to do single-step forecasting on the returns of each stock. For each variable, we try its various values with all other variables's values fixed. We repeat the experiment **20** times and report the loss on the out-of-sample test dataset to represent its generalization error. We also obtain a standard deviation of the generalization error. We use mean square error (MSE) as evaluation metrics.

To show our theory's generality on time series prediction problems, we also conduct the same set of experiments on a second multivariate time series dataset from a different domain. We use the Beijing PM2.5 dataset from UCI [3]. It is a by-hour dataset. The prediction target is the PM2.5 concentration in real numbers. The task is to use 7 features' values in the past 7 hours to predict the PM2.5 concentration value in the next hour.

Table 4: Experiment environment. The list only includes major packages. All the packages are installed using Anaconda and Pip.

Package	Version
python	3.9.13
matplotlib	3.5.2
numpy	1.23.1
pandas	1.4.4
pytorch	1.12.1
torch-geometric	2.1.0.post1
torch-scatter	2.0.9
torch-sparse	0.6.15

E.3 Neural Network Architecture

We implement our neural network as a three-layer Graph Attention Neural network. This includes the input layer, one hidden layer, and the output layer. Each layer is a GATConv layer from the Pytorch-Geometric package [4]. We use ELU activation (Clevert et al., 2015) and Dropout (Srivastava et al., 2014) after both the input layer and the hidden layer.

For the number of heads variable, we change the number of attention heads of each layer and all layers use the same number of attention heads. For the number of neighbors variable, we adjust the dropout rate inside each layer's attention mechanism (different from the dropout layer) so that only a percentage of the nodes are considered when using the attention to aggregate information from a node's neighbors. For the weight norm variable, we adjust the bound of the Frobenius norm of the weight matrix inside each layer by using weight clipping. Each element of the weight matrix is clipped to the threshold to make sure the Frobenius norm of the matrix is less than or equal to the bound. For the number of layers variable, we adjust the number of hidden layers ranging from 1 to 8. For the input norm variable, we make sure each node's feature vector's norm is less than or equal to a bound ranging from 1 to 28. For the maximum number of neighbors, we first create a graph network from the data. We calculate pairwise correlations of nodes using the features, forming a correlation matrix. For each node, we select the top N_e nodes with the strongest correlations as its neighbors.

[2] `https://www.kaggle.com/datasets/paultimothymooney/stock-market-data`
[3] https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data
[4] `https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html`

E.4 Model Hyperparameters

Table 5: Default hyperparameters. When we study the impact of different values of a variable such as the number of heads, we keep all other variables fixed to the a set of same values.

Hyperparameter	Value	Comment
num-hid-layers	1	Number of hidden GATConv layers
num-heads	2	Number of attention heads; Same for all layers
num-neighbors	0.1	Neighbors to obtain attentions; Percentage of all nodes
train-ratio	0.6	Ratio of training dataset
inputs-norm	1.0	Bound of the norm of the inputs
weights-bound	None	Bound of the norm of the model weights if not None
hidden-size	32	Out-channels of the input layer and hidden layer
lr	1e-4	Learning rate
dropout	0.1	Dropout rate after input layer and hidden layer

We provide source code for the reproducibility of the paper. The code is posted on GitHub[5].

E.5 Best Model Hyperparameters

Below we present the best hyperparameters used in our method.

Table 6: Best hyperparameters.

Hyperparameter	Value	Comment
num-hid-layers	2	Number of hidden GATConv layers
num-heads	16	Number of attention heads; Same for all layers
num-neighbors	6	Number of neighbors to obtain attentions;
inputs-norm	1.0	Bound of the norm of the inputs
weights-bound	4.0	Bound of the norm of the model weights if not None
hidden-size	32	Out-channels of the input layer and hidden layer
lr	1e-4	Learning rate
dropout	0.1	Dropout rate after input layer and hidden layer

E.6 Computational Complexity

Computational Complexity: The time complexity of the L layers GAT with K attention heads can be expressed as $\mathcal{O}(KL(NDC + |E|C))$. Here, D represents the number of input features, C is the number of output features, and N and $|E|$ denote the numbers of nodes and edges in the graph, respectively. This calculation is based on the reference in Veličković et al. (2017).

F More Experiments And Discussions To Verify The Theoretical Bounds

F.1 Discussions About Results For Three-Layer GATs

In our experiment, we use a three-layer GAT to demonstrate Theorem 4.1, Figure 1 shows that our experiment result is consistent with the theory. Below we provide more discussions about the results.

Number of Attention Heads - K. For three-layer GATs, Theorem 4.1 indicates that with the increasing number of attention heads in the attention layer, the upper bound of ERC is $y = O(K^{L-1})$. The experiment results in Figure 1 shows test error beginning to increase quadratically after some values of K. This is consistent with our theoretical results on the generalization error bound.

[5] `https://github.com/zzh237/Weight-bounded-GAT`

Maximum Number of Neighbors - N_e. For the maximum number of neighbors, the theoretical upper bound of ERC is $y = \mathcal{O}(N_e^L)$. Figure 1 demonstrates that as the N_e increases, the test error conforms to this theoretical error bound. But it is noteworthy that when the number is too small, the loss is also high. It is possible that at a certain range, the influence of the information loss due to the limited number of neighbors is dominant. a node may only depend on a limited number of other nodes, and increasing its neighborhood could introduce noises to the model for MTSF.

Norm of Weight Matrix - M_l. Theorem 4.1 indicates that with the increasing bound of the Frobenius norm, the upper bound of ERC increases polynomially. The experiment results in Figure 1 corroborate the conclusion. When the weight norm increases, the generalization error first decreases, then increases. The reason for the initial decrease is because the bound on the norms is so small that it severely prevents the weights from having enough amount of updates. Thus, the scales (norms) of the weight matrices should be neither too large (induces large generalization error) nor too small (harms weights' updates) and choosing proper scales is important in practice.

Norm of Inputs - B. The experiment results in Figure 1 show that when the input norm is greater than certain values, the generalization loss of the GAT has a linear relationship with the upper bound of the input norm. This empirical observation aligns with Theorem 4.1.

Sample Size - n. Our Theorem 3.1 reports that the ERC has a polynomial $\mathcal{O}(n^{-1/2})$'s dependence on the sample size. As we can see from Figure 1, the generalization loss decreases when the training data set size increases. The red curve based on the theoretical value matches the pattern of the blue and yellow curves based on our experiments.

F.2 For Two-Layers

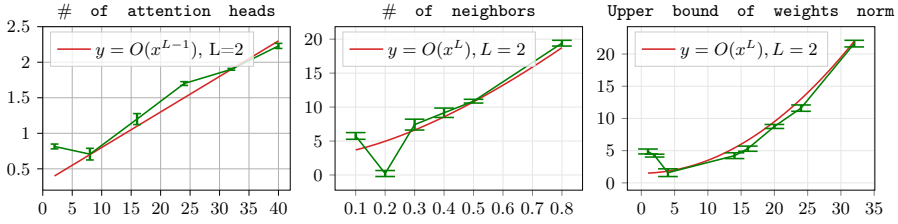

Figure 2: Additional experiment results on the stock dataset using a two-layer GAT on three variables in the ERC. We run the experiment 20 times and obtain a standard deviation of the generalization error. (a) relationship between test loss and different variables. The red line is a possible theoretical upper bound. The plots show that when L equals to 2, all test losses still generally conform to the big $\mathcal{O}$ of the theoretical upper bound.

In the experiment section, for all experiments, where the number of layers, L, needs to be fixed for studying the relationship of the generalization error with different GAT components that impact the ERC, including the number of attention heads, the number of neighbors, the upper bound of weights norm.

However, the upper bound of input norm and the training data size affecting the ERC do not depend on the number of layers, thus, in our supplemental experiment for two-layer GATs, we only consider the three variables (the number of attention heads, the number of neighbors, and the upper bound of weights norm) that impact ERC and depend on the number of layers. Notably, Theorem 3.1 for generalization error bound of two-layer GATs is just a special case of Theorem 4.1 for deep GATs: when we let L equal to 2 in Theorem 4.1, the generalization error bound is identical to the bound in Theorem 3.1 multiplied with some constants. We now give empirical results on two-layer GATs to see if it is consistent with Theorem 3.1 and Theorem 4.1.

In this appendix section, we include more experiment results from a two-layer GAT on three variables in the ERC on the stock dataset. As Figure 2 shows, when the number of attention heads increases, the generalization error increases at a linear rate after certain values. When the number of neighbors increases, the generalization error initially decreases when the number of neighbors is small, then starts increasing following the trend of a polynomial function. As for the upper bound of weight norm, the generalization error increases quadratically with the increment of the norm. The empirical results for all three variables generally conform to the theoretical bound suggested by Theorem 3.1 as well as Theorem 4.1. For some inconsistency between the results and the reference red line or theoretical results at the the beginning, the reason, as we have discussed, can be due to

the trade-off between approximation error and estimation error, since when the complexity of the hypothesis class increases, the former decreases and the latter increases. In the long run, the estimation error or the ERC dominates the bound.

Thus, the results for $L = 2$ along with the three-layer experiments provide evidence for our theoretical findings (Theorem 3.1 and Theorem 4.1).

G Experiments Of Comparing the Weights-bounded GAT with Weight decay

we conducted an additional hyperparameter search experiment to compare the Weight-bounded GAT with Vanilla GAT.

The experiment was repeated five times to account for variations with different random seeds. For each random seed, for Vanilla GAT, the weight decay hyperparameter was tuned, while for Weight-bounded GAT, the weight bound was tuned and the weight decay was not used (set to 0). The remaining hyperparameters were kept the same. The testing Mean Squared Error (MSE) and standard deviation across different random seeds are presented in the subsequent table.

For hyperparameter tuning, we employed the Bayesian optimization method using Hyperopt http://hyperopt.github.io/hyperopt/. For each random seed, the optimization process ran 50 trials. The weight decay value in each trial was sampled from a lognormal distribution $\lambda \sim \text{Lognormal}(1, 1)$. The weight bound value in each trial was sampled from a lognormal distribution $M \sim \text{Lognormal}(1, 1)$.

Table 7: Comparion of Weights-bounded GAT with Vanilla GAT with Tuned Weight Decay

Dataset	Method	Mean(10^{-3})	Std	seed1	seed2	seed3	seed4	seed5
Stock	Weight-bounded GAT	0.531	0.004	0.531	0.527	0.534	0.526	0.536
	Vanilla GAT With Tuned Weight Decay	2.641	3.920	0.622	1.630	0.544	0.800	9.610
Pm25	Weight-bounded GAT	78.98	0.821	79.75	79.18	77.83	78.48	79.68
	Vanilla GAT With Tuned Weight Decay	81.72	1.343	81.49	80.71	83.63	79.95	82.80

The results table shows that it is still true that the Weight-bounded GAT with fine-tuned weight bound outperforms the Vanilla GAT with fine-tuned weight decay. The Weight-bounded GAT with tuned weight bound has both smaller average MSE across different random seeds and smaller standard deviation. While the Vanilla GAT with tuned weight decay would not achieve the same performance and it also has more variation across different random seeds. The result is consistent with our intial findings in the paper.

H Comparison With Another Method For MTSF

In this section, to further evaluate our method, we also compare the performance of the Weights-bounded GAT against a top-tier time series forecasting model that is not based on graph neural networks, the Patch Time Series Transformer (PatchTST)Nie et al. (2022).

Table 8 presents the results, indicating that our method outperforms PatchTST across different prediction horizons on eight common multivariate time series datasets.

Table 8: Comparison with PatchTST on multiple datasets

	Prediction horizon (ETTh1)				Prediction horizon (Traffic)				Prediction horizon (ILI)				Prediction horizon (ETTm1)			
	96	192	336	720	96	192	336	720	24	36	48	60	96	192	336	720
Ours	0.354	0.392	0.406	0.430	0.345	0.362	0.374	0.419	1.303	1.560	1.535	1.447	0.289	0.329	0.363	0.409
PatchTST	0.37	0.413	0.422	0.447	0.360	0.379	0.392	0.432	1.319	1.579	1.553	1.470	0.293	0.333	0.369	0.416
	Prediction horizon (Weather)				Prediction horizon (Electricity)				Prediction horizon (ETTh2)				Prediction horizon (ETTm2)			
	96	192	336	720	96	192	336	720	96	192	336	720	96	192	336	720
Ours	0.131	0.180	0.228	0.297	0.121	0.145	0.156	0.191	0.267	0.328	0.317	0.350	0.164	0.221	0.271	0.349
PatchTST	0.149	0.194	0.245	0.314	0.129	0.147	0.163	0.197	0.274	0.341	0.329	0.379	0.166	0.223	0.274	0.362

I Overview of Baselines used in Table 1 For MTSF

In this section, we provide an overview of the baselines we used. GDN is proposed by Deng & Hooi (2021) and is used for anomaly detection in multivariate time series, however, it is originally used for classification. To adapt it to our settings, we modify the output layer of GDN to perform the regression instead of the classification

task. Apart from using graph attention mechanism, GDN also learns a directed graph to model the causal-effect relationship between different nodes, which can capture the asymmetric dependency patterns. ASTGCN is developed by Guo et al. (2019) for multiple traffic flow forecasting. ASTGCN considers the attention along the time axis and between nodes. GMAN is proposed by Zheng et al. (2020) for traffic prediction. GMAN uses a multi-attention mechanism to capture the dependencies among different graph regions and also employs a graph pooling layer to adaptively select the most informative regions. However, the mentioned attention-based methods may be prone to overfitting, especially when their weight matrix norm bounds are large, along with a high number of attention heads and a large maximum number of neighbors. Our analysis indicates that the generalization error increases if the weight matrix norm's upper bound is large, the number of attention heads is high, and the maximum number of neighbors is extensive.

For other methods that not consider attention mechanisms, the STGCN (Yu et al., 2017) employs 1D convolution filters on the temporal dimension to capture temporal dependencies. MTGNN(Zhang et al., 2020) operates on a hierarchical graph structure, where multiple graphs are stacked to capture the dependencies at different levels of granularity. The attention mechanism could further be used to capture dependencies.

As shown in Table 1, the weight-bounded GAT performs better than all the baselines, including the vanilla GAT that does not add the weight bound. Our theoretical framework, which explores the relationship between the generalization error bound and the weight matrix norm bound, supports these experimental findings.