Unified Insights into Graph Neural Networks: From Connectivity to Linear Architectures, and Time Series Applications

PRATHAM TANEJA[1], SANSKAR SAXENA[2], DIVYANSH SINGH[3], ADITYA CHAUHAN[4], BHAUMIK TYAGI[5], GARV KALIA[6]

[1] *Graduate Student,(Electronics and Communication Engineering), ADGITM, Delhi, India*

[2] *Undergraduate Student,(Computer Science Engineering), Moradabad Institute of Technology, Uttar Pradesh, India*

[3] *Undergraduate Student,(Information Technology), ADGITM, Delhi, India*

[4] *Undergraduate Student,(Computer Science Engineering), Moradabad Institute of Technology, Uttar Pradesh, India*

[5] *Jr. Research Scientist,(Computer Science Engineering), Delhi, India*

[6] *Undergraduate Student,(Information Technology), ADGITM, Delhi, India*

Abstract- ***The predominant data modality employed for the documentation of dynamic system measurements is time series, which emanates prolifically from physical sensors and online processes (virtual sensors). The imperative role of time series analytics in extracting the inherent wealth of information from available data is underscored by recent advancements in graph neural networks (GNNs). These networks have witnessed a notable surge in their application for time series analysis, owing to their capacity to explicitly model inter-temporal and inter-variable relationships—attributes that traditional and other deep neural network-based methodologies find challenging. This review endeavors to comprehensively review graph neural networks for time series analysis (GNN4TS), encompassing four fundamental dimensions: forecasting, classification, anomaly detection, and imputation. The overarching objective is to serve as a guiding resource for designers and practitioners, facilitating an enhanced understanding, application development, and progression of research within the domain of GNN4TS.Linear Architectures are also discussed in this very research. Neural Architecture Search (NAS) has exhibited notable advancements in optimizing Graph Neural Networks (GNNs), denoted as NAS-GNNs, outperforming manually designed GNN architectures. Nevertheless, challenges inherited from conventional NAS methods, including elevated computational costs and optimization complexities, persist. Notably, prior NAS approaches have tended to overlook the inherent characteristics of GNNs, which inherently possess expressive power without the necessity for training. Notably, NAC achieves up to a 200× acceleration in computational efficiency and a 19.9% enhancement in accuracy compared to robust baseline methods.***

Indexed Terms- ***Graph neural networks, Linear architecture search, Time series Applications.***

I. INTRODUCTION

In recent years, the application of graph neural networks (GNNs) has emerged as a potent methodology for acquiring non-Euclidean data representations [1], facilitating the modeling of real-world time series data. This approach proves instrumental in capturing intricate and diverse relationships, encompassing both inter-variable connections within multivariate series and inter-temporal dependencies across different temporal points. Recognizing the inherent spatial-temporal complexities within real-world scenarios, a body of research has amalgamated GNNs with diverse temporal modeling frameworks to encapsulate both spatial and temporal dynamics, yielding promising outcomes [2].

While initial investigations predominantly focused on various forecasting scenarios [3], recent strides in time series analysis leveraging GNNs have exhibited favorable results across other key tasks, including classification [4], anomaly detection [5], and imputation [6].The proliferation of advanced sensing technologies and data stream processing has engendered an unprecedented influx of time series data [7]. The analysis of time series data not only facilitates retrospective trend identification but also underpins a spectrum of tasks such as forecasting [8], classification [9], anomaly detection [10], and data imputation [11]. This substantiates the foundation for time series modeling paradigms that leverage historical data to comprehend present and future possibilities. Time series analytics has assumed heightened significance across diverse domains, spanning cloud computing, transportation, energy, finance, social networks, and the Internet-of-Things [12].

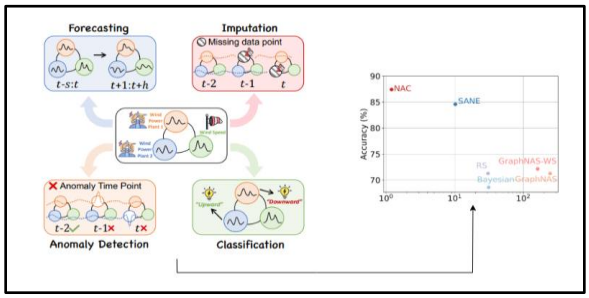

Fig. 1: Graph neural networks for time series analysis (GNN4TS) and neural linear architecture search.

II. LITERATURE REVIEW

Despite the burgeoning body of research dedicated to various time series analytic tasks employing Graph Neural Networks (GNNs), extant surveys often exhibit a proclivity for specific perspectives within delimited scopes. Notably, the survey conducted by Wang et al. [13] provides a comprehensive review of deep learning techniquesfor spatial-temporal data mining; however, its focus is not specifically directed towards GNN-based methodologies. Similarly, the survey conducted by Ye et al. [14] delves into graph-based deep learning architectures within the traffic domain, with a primary emphasis on forecasting scenarios. A more recent survey by Jin et al. [15] furnishes an overview of GNNs applied to predictive learning in urban computing; however, it confines its coverage to this application domain and does not thoroughly address other tasks pertinent to time series analysis. To address the identified gap in the existing literature, the present survey endeavors to furnish a comprehensive and contemporaneous examination of graph neural networks applied to time series analysis. Encompassing a spectrum of tasks, including time series forecasting, classification, anomaly detection, and imputation, this survey adopts a systematic approach. The initial phase involves the presentation of two overarching perspectives for the classification and discussion of extant works, focusing on both task and methodology-oriented considerations.

Subsequently, the survey delves into an exploration of six prominent application sectors within the domain of Graph Neural Networks for Time Series Analysis (GNN4TS). To further contribute to the scholarly discourse, the survey concludes by positing several potential future research directions. The key contributions of this survey are succinctly summarized as follows:

- This survey stands as the inaugural comprehensive examination, systematically reviewing recent advancements in mainstream time-series analysis tasks employing graph neural networks. It spans a diverse array of recent research initiatives, presenting an extensive perspective on the evolution of Graph Neural Networks for Time Series Analysis (GNN4TS) without confining its scope to specific tasks or domains.
- This study presents a unified framework for the structural categorization of existing works within the domain of Graph Neural Networks for Time Series Analysis (GNN4TS), employing a dual perspective based on tasks and methodologies. The first classification provides a comprehensive overview of tasks inherent in time series analysis, addressing diverse problem settings prevalent in GNN-based research. The second classification dissects GNN4TS with a focus on spatial andtemporal dependencies modeling, alongside an in-depth exploration of the overall model architecture.
- A thorough review is conducted to illuminate the Linear Architecture implemented through GNN,

ensuring comprehensive coverage of the breadth of the field. This review extends beyond a superficial examination, incorporating fine-grained classification and detailed discussions to provide readers with an updated comprehension of the current state-of-the-art in GNN4TS.

- The study additionally explores the expanding applications of GNN4TS across various sectors, emphasizing its versatility and potential for future growth in diverse fields. Furthermore, the research sheds light on prospective directions for future research, offering insights and suggestions that may serve as guidance and inspiration for subsequent investigations within the field of GNN4TS.

III. RELATED BACKGROUND

Time series data encompasses a sequence of observations congregated or documented over a period. This data can beeither *regularly* or *irregularly sampled*, with the latter alsoreferred to as time series data with missing values. Withineach of these cases, the data can be further classified intotwo primary types: *univariate* and *multivariatetime series*. Inthe sequel, we employ bold uppercase letters (e.g., X), boldlowercase letters (e.g., x), and calligraphic letters ($e.g., \mathcal{V}$) todenote matrices, vectors, and sets, respectively.Most of the research grounded on GNNs concentrates onmodeling multivariate time series, as they can be naturally abstracted into *spatial-temporal graphs*. This abstractionallows for an accurate characterization of dynamic inter-temporal and inter-variable dependencies. The former describes the relations between different time steps withineach time series (e.g., the temporal dynamics of red nodesbetween *t1* and *t3* in Fig. 2), while the latter capturesdependencies between time series (e.g., the spatial relationsbetween four nodes at each time step in Fig. 2), such as thegeographical information of the sensors generating the datafor each variable. To illustrate this, we first define attributedgraphs.

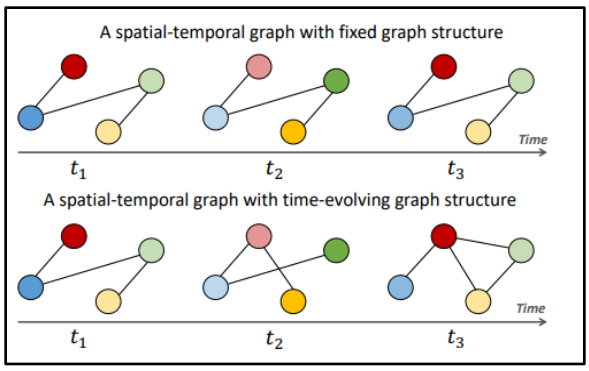

Fig. 2: Illustrations of spatial-temporal graphs.

Considering, a spatial-temporal graph can be described as a series of attributed graphs, which effectively represent (multivariate) time series data in conjunction with either evolving or fixed structural information over time.

Graph neural networks (GNNs) are introduced as contemporary deep learning models designed for the processing of *graph-structured data*. The central operation within conventional GNNs, commonly denoted as *graph convolution*, entails the exchange of information among neighboring nodes. In the realm of time series analysis, this operation facilitates the explicit consideration of inter-variable dependencies as delineated by the graph edges. Cognizant of the varied intricacies, GNNs are characterized within the spatial domain, entailing the transformation of the input signal through learnable functions along the N-dimensional dimension.

GNN Definition: - Given an attributed graph G = (A, X), we define x_i = X[i,:] $\in \mathbb{R}^D$as the D-dimensional feature vector of node v_i. A GNN learns node representations through two primary functions: AGGREGATE (·) and COMBINE (·).

The AGGREGATE (·) function computes and aggregates messages from neighboring nodes, while the COMBINE (·) function merges the aggregated and previous states to transform node embeddings. Formally, the *k*-th layer in a GNN is defined by the extended

$a_i^{(k)}$= AGGREGATE$^{(k)}$ ({$h\_j\hat{}((k-1)): v\_j \in \mathcal{N}(v\_i)$}),
$h_i^{(k)}$ = COMBINE$^{(k)}$ 〖(h〗 _$i\hat{}((k-1))$, $a_i^{(k)}$), (1)

or, more generally, aggregating messages computed from both sending and receiving nodes v_j and v_i, respectively. Here,$a_i^{(k)}$and $h_i^{(k)}$represent the aggregated message from neighbors and the transformed node embedding of node v_i in the k-th layer, respectively. The input and output of a GNN are $h_i^{(0)}$: = x_i and $h_i^{(K)}$: =h_i.

IV. METHODOLOGY

Time-Series Applications using Graph Neural Networks:

Time series analysis using Graph Neural Networks (GNNs) has garnered considerable attention due to its capacity to model complex relationships and dependencies within temporal data. The application of GNNs in this context extends across various domains, showcasing their versatility and efficacy. Here, we explore a few key applications of GNNs in time series analysis:

1. Time Series Forecasting: GNNs excel in capturing intricate temporal dependencies, making them well-suited for time series forecasting. By leveraging the inherent graph structure of time series data, GNNs can discern patterns and trends, enhancing the accuracy of predictions. This application is particularly valuable in domains such as finance, weather prediction, and energy consumption forecasting.
2. Anomaly Detection: Identifying anomalies in time series data is crucial for various industries, including cyber security, healthcare, and manufacturing. GNNs can effectively detect unusual patterns by learning the normal behavior of the time series data. Anomalies, represented as deviations from the learned patterns, can be flagged for further investigation.

3. Classification in Time Series Data: GNNs offer a powerful approach to classifying time series data into different categories. This is valuable in scenarios where the temporal dynamics of the data play a significant role in determining its class. Applications include activity recognition in motion sensor data, event detection in social networks, and disease classification in healthcare.

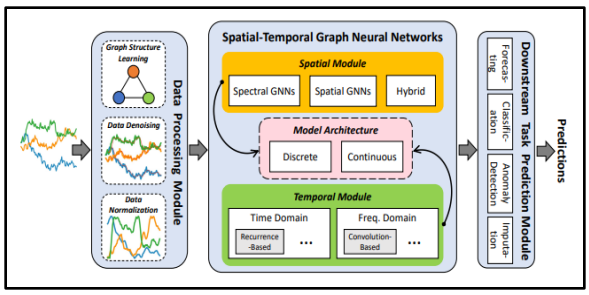

Fig 3: Wide-ranging pipeline for time series analysis using Gnn's

4. Imputation of Missing Data: Time series data often suffer from missing values, which can hinder the analysis. GNNs can be employed to impute missing data points by leveraging information from neighboring time points. This is beneficial in domains like finance, where incomplete historical stock prices or economic indicators may impede accurate analysis.

5. Dynamic Graphs for Evolving Time Series: Many real-world systems exhibit dynamic changes over time, leading to variations in the underlying relationships within the time series data. GNNs with dynamic graph structures can adapt to these changes, making them suitable for applications in dynamic social networks, evolving financial markets, and changing environmental conditions.

- Time Series Forecasting: -

This task is centered around predicting future values of the time series based on historical observations, as depicted in Fig. 4a. Depending on application needs, we categorize this task into two types: single-step-ahead forecasting and multi-step-ahead forecasting. The former is meant to predict single future observations of the time series once at a time, i.e., the target at time t is$Y$: = $X_{t+H}$for some $H\in\mathbb{N}$ steps ahead, while the latter makes predictions for a time interval, e.g., $Y := X_{t+1:t+H}$. Parameterized solutions to both predictive cases can be derived by optimizing,

$\theta^{*}, \varphi^{*} = \arg\min \mathcal{L}\_F\ (p\varphi(f\_\theta\ (X\_(t-T:t), A\_(t-T:t)\ )), Y)$, (2)

where $f\theta(\cdot)$ and $p\varphi\ (\cdot)$represent a spatial-temporal GNN and the predictor, respectively. In the sequel, we denote by $X_(t-T:t)$and $A_(t-T:t)$a spatial-temporal graph with length.

- Time Series Imputation: -

This task is centered around estimating and filling in missing or incomplete data points within a time series. Current research in this domain can be broadly classified into two main approaches: in-sample imputation and out-of-sample imputation. In-sample imputation involves filling missing values in a given time series, while out-of-sample imputation pertains to inferring missing data not present in the training dataset. We formulate the learning objective as follows:

$\theta^{*}, \varphi^{*} = \arg\min \mathcal{L}_I\ (p\varphi(f_\theta\ (\tilde{X}_(t-T:t), A_(t-T:t)\ )), X_(t-T:t)\ ),$ (3)

where $f\theta(\cdot)$ and $p\varphi\ (\cdot)$denote the spatial-temporal GNN and imputation module to be learned, respectively. The imputation module can e.g., be a multi-layer perceptron. In this task, $\tilde{X}_(t-T:t)$represents input time series data with missing values (reference time series), while $X_(t-T:t)$denotes the same time series without missing values.

- Time Series Classification: -

This task aims to assign a categorical label to a given time series based on its underlying patterns or characteristics. Rather than capturing patterns within a time series data sample, the essence of time series classification resides in discerning differentiating patterns that help separate samples based on their class labels. The optimization problem can be expressed as:

$\theta^{*}, \varphi^{*} = \arg\min \mathcal{L}_C\ (p\varphi(f_\theta\ (X, A)), Y),$ (4)

where $f\theta(\cdot)$ and $p\varphi\ (\cdot)$denote, e.g., a GNN and a classifier to be learned, respectively. Using univariate time series classification as an example, the task can be formulated as either a graph or node classification task. In the case of graph classification *(Series-As-Graph)*[16], each series is transformed into a graph, and the graph will be the input of a GNN to generate a classification output. This can be achieved by dividing a series into multiple subsequence's with a window size, W, serving as graph nodes, $X \in \mathbb{R}^{N \times W}$, and an adjacency matrix, A, describing the relationships between subsequence's.

- Time Series Anomaly Detection: -
- This task is centered on the identification of irregularities and unexpected events in time series data. The process of anomaly detection entails both determining when anomalous events occurred and gaining insights into the underlying factors contributing to the anomalies. Given the inherent challenges associated with acquiring labeled anomaly events, contemporary research predominantly treats anomaly detection as an unsupervised problem. This involves designing a model that characterizes normal, non-anomalous data. Subsequently, the trained model is deployed to detect anomalies by generating elevated scores when anomalous events manifest.
- The model learning process aligns with the optimization for forecasting, as denoted by Eq. 2, wherein fϑ(·) and pϕ(·) represent the spatial-temporal Graph Neural Network (GNN) and the predictor, respectively. Typically, both the spatial-temporal GNN and the predictor are trained on normal, non-anomalous data, employing forecasting [17] or reconstruction [18] optimization methodologies. The objective is to minimize the discrepancy between the normal input and the forecasted (or reconstructed) series. However, when these models are tasked with anomaly detection, their failure to conform to expected low-discrepancy behavior during anomaly periods creates discernible differences, facilitating anomaly detection.
- The determination of the threshold demarcating normal and anomalous data represents a crucial hyperparameter, necessitating consideration of the rarity of anomalies and alignment with a specified false alarm rate [19]. Finally, for diagnosing the causes of anomalies, a prevalent strategy involves computing discrepancies for each channel node and consolidating these into a singular anomaly score [20]. This approach enables the identification of channel variables responsible for anomaly events by assessing their respective contributions to the final anomaly score.

The Spatial Module of Spatio-Temporal Graph Neural Networks (STGNNs) is designed to model dependencies between time series over time, drawing inspiration from the principles of Graph Neural

Networks (GNNs) applied to static graphs. This module encompasses three distinct types: spectral GNNs, spatial GNNs, and a hybrid combination of both. Spectral GNNs leverage spectral graph theory and utilize graph shift operators, such as the graph Laplacian, to capture node relationships in the graph frequency domain[21]. In contrast, spatial GNNs simplify spectral GNNs by directly designing filters localized to each node's neighborhood. Hybrid approaches amalgamate both spectral and spatial methodologies to harness the strengths of each method[22].

The Temporal Module is introduced in STGNNs to account for temporal dependencies within time series data.Temporal dependencies can be represented in either the time or frequency domains. In the first category, methods include recurrence-based approaches (e.g., RNNs), convolution-based methods (e.g., Temporal Convolutional Networks - TCNs), attention-based techniques (e.g., Transformers), and hybrid combinations of these. The second category employs analogous techniques, incorporating orthogonal space projections like the Fourier transform[23].

Concerning Model Architecture, existing STGNNs can be classified as either discrete or continuous in terms of their overall neural architectures. These types further categorize into factorized and coupled subcategories[24]. Factorized STGNN model architectures entail temporal processing either before or after spatial processing, either discretely (e.g., Spatio-Temporal Graph Convolutional Network - STGCN) or continuously (e.g., Spatio-Temporal Graph Ordinary Differential Equation - STGODE). Conversely, coupled model architectures refer to instances where spatial and temporal modules are interleaved, exemplified by models such as Discrete ChebNet Recurrent Neural Network (DCRNN) and Continuous MTGODE. Alternative nomenclature designates these categories as time-then-space and time-and-space architectures [25].

- Linear Architecture Search:

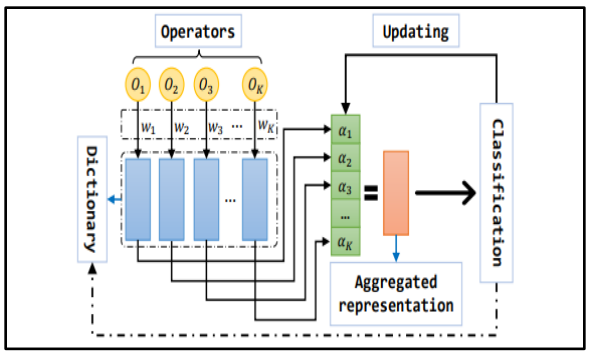

Fig 4: The proposed NAC architecture in one layer.

With a fixed dictionary, NAC learns the architecture immediately. NAC-updating refreshes the dictionary in training, with a dashed line indicating the additional step.

Algorithm:The NAC algorithm

Require: The search space $\mathcal{A}$;
Ensure: The architecture α
Randomly initializingW^l, for l = 1, . . . , L; set α = 1;
1: while t=1,..,T do
2: Performing the feature aggregation at each layer aso^{-1} (x)= o^l $\hat{\alpha}_1 = \sum_{k=1}^{K} \frac{\alpha_{lk}}{\|\alpha\|_2} o^{lk}(x)$;
3: Computing$h_v^l = \emptyset\left[W^l \cdot 0^{-l} \cdot \left(\{h_u^{l-1}, \forall u \in \mathcal{N}(v)\}\right)\right]$;
4: Optimizingαbased on the objective function;
5: UpdatingW_0based on the objective function;
6: end while
7: Obtain the final architecture {α*} from the trainedα via an argmax operation at each layer;

In this work, we propose two realizations of NAC, namely NAC and NAC-updating. The computation of NAC has two major parts: the forward pass and the backward pass. Given the search space, the computation of the forward is then fixed, regarding as a constant. Therefore, the computational complexity mainly focuses on the backward pass in the NAC algorithms.

The main version of our work has no need to update weights, but only to update architectural parameter α during the training process. Therefore, the algorithmic complexity is as *O (T ∗‖α‖)*, which is a linear function w.r.t *α*. The dimension of *α* is often

small, which makes the model easy to scale to large datasets and high search space. It is called'NAC'.

When updating weights, like DARTS, the complexity is estimated as $O(T * (||\alpha|| + ||w||))$. The dimension of w is often much larger than α, therefore, the complexity is dominated by updating w, where the complexity is $O(T * ||w||)$. Since the dimension of α is much smaller than w, the complexity of NAC is much less than this version.

Searching configuration is presented as:

1. *Architecture optimizer:* Adam is used for training the architecture parameters α. The learning rate is set as 0.0004 and the weight decayas 0.001. Also, the β1 and β2 are fixed as 0.6 and 0.999, respectively. All runs a constant schedule for training, such as 100 epochs.
2. *Weight optimizer:* SGD is used to update models' parameters, i.e., w. The learning rate and SGD momentum are given as 0.027 and 0, respectively, where the learning rate has a cosine decay schedule for 100 epochs. The weight decay value is fixed as, i.e., set ρ1 = 0.0006.
3. *Batch size:* For transudative tasks, in-memory datasets are utilized, and the batch size is fixed as the size of the dataset themselves.

V. DISCUSSION & ANALYSIS

Time series classification task seeks to assign a categorical label to a given time series based on its underlying patterns or characteristics. By transforming time series data into graph representations, one can leverage the powerful capabilities of GNNs to capture both local and global patterns. Furthermore, GNNs are capable of mapping the intricate relationships among different time series data samples within a particular dataset.

Table 1: GNNs for Time Series Classification

Approach	Task	Spatial Module	Temporal Module	Conversion	Graph Heuristics
LB-Sim	Univariate+ Multivariate	Spatial	Time Domain-Convolution	Series-As-Node	Pairwise Similarity
TSC [42] 2023	Time series classification	GNN	in-Convolution	Node	Similarity
Sim TSC [43] 2022	Univariate+ Multivariate Time series classification	Spatial GNN	Time Domain-Convolution	Series-As-Node	Pairwise Similarity

The advent of deep learning has sparked significant advancements, drawing lessons from earlier methods. Early research in this area proposed recurrent models with reconstruction, and forecasting [26] strategies respectively to improve anomaly detection in multivariate time series data. The forecasting and reconstruction strategies rely on forecast and reconstruction errors as discrepancy measures between anticipated and real signals. These strategies rely on the fact that, if a model trained on normal data fails to forecast or reconstruct some data, then it is more likely that such data is associated with an anomaly. However, recurrent models [27] are found to lack explicit modeling of pairwise interdependence among variable pairs, limiting their effectiveness in detecting complex anomalies [28]. Recently, GNNs have shown promising potential to address this gap by effectively capturing temporal and spatial dependencies among variable pairs [29].

Table 2: GNNs for Time Series Anomaly Detection

Approach	Architecture	Spatial Module	Temporal Module	Learned Relations	Graph Heuristics
DyGraphAD [44] 2023	Forecast+Relational Discrepancies	Spatial GNN	Time Domain-Convolution	NA	Pairwise Similarity
CST-GL [45] 2023	Forecast	Spatial GNN	Time Domain-Convolutio	Static	NA

			n		
Grelen [46] 2022	Reconstruction+Relational Discrepancies	Spatial GNN	Time Domain-Hybrid	Dynamic	NA

Time series imputation, a crucial task in numerous real-world applications, involves estimating missing values within one or more data point sequences. Traditional time series imputation approaches have relied on statistical methodologies, such as mean imputation, spline interpolation [30], and regression models [31]. However, these methods often struggle to capture complex temporal dependencies and non-linear relationships within the data. While some deep neural network-based works, such as [32],have mitigated these limitations, they have not explicitly considered inter-variable dependencies. The recent emergence of graph neural networks has introduced new possibilities for time series imputation. From a task perspective, GNN-based time series imputation can be broadly categorized into two types: in-sample imputation and out-of-sample imputation. The former involves filling in missing values within the given time series data, while the latter predicts missing values in disjoint sequences.

Table 3: GNNs for Time Series Imputation

Approach	Task	Spatial Module	Temporal Module	Type	Graph Heuristics
GARNN [47]2023	In-sample	Spatial GNN	Time Domain-Recurrence	Deterministic	Pairwise Connectivity
DGCRIN [48] 2023	In-sample	Spatial GNN	Time Domain-Recurrence	Deterministic	Pairwise Similarity

Time series forecasting aims to predict future time series values based on historical observations. In recent years, deep learning-based approaches have demonstrated considerable success in forecasting time series by capturing non-linear temporal and spatial patterns more effectively than the linear counterpart [33]. Techniques such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and attention-based neural networks have been employed. However, many of these approaches, such as LSTNet [34] and TPA-LSTM [35], overlook and implicitly model the rich underlying dynamic spatial correlations between time series. Recently, graph neural network (GNN)- based methods have shown great potential in explicitly and effectively modeling spatial and temporal dependencies in multivariate time series data, leading to enhanced forecasting performance.

Table 4: GNNs for Time Series Forecasting

Approach	Architecture	Spatial Module	Temporal Module	Learned Relations	Graph Heuristics
Jin et al. [49] 2023	Discrete-Factorized	Spectral GNN	Frequency domain-Hybrid	Static	NA
SGP [50] 2023	Discrete-Factorized	Spatial GNN	Time Domain-Recurrence	NA	Spatial Proximity, Pairwise Similarity
RGSL [51] 2022	Discrete-Continuous	Spectral GNN	Time Domain-Recurrence	Static	Spatial Proximity, Pairwise Connectivity

The effect of training on the final linear layer:
Proposed theorems prove that a GNN with randomly initialized weights can make the final output as good as a well-trained network when initializing networks with orthogonal weights and updating the total network using gradient descent. In practice, we find it

difficult to determine at what training epoch the optimal weight parameters can be obtained through training linear layer. We noticed that most of the time, the untrained weights in the initial state can often already exceed the accuracy that can be obtained from the weights after multiple epochs of training the final linear layer, as shown

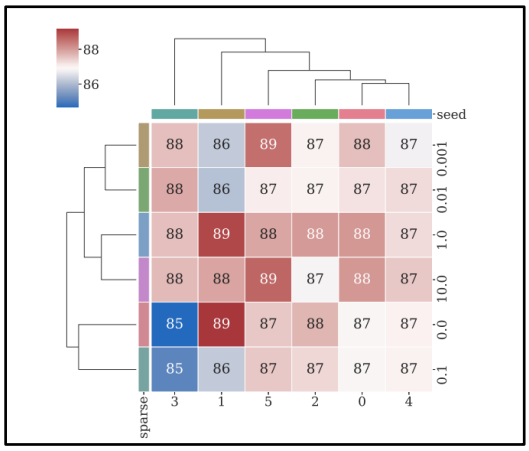

Figure 6: The effects of random seeds of NAC on the accuracy, where x-axis denotes the random seeds and y-axis denotes the sparsity. NAC performs stably with random seeds.

Therefore, we further omit the training of the final linear layer. It is important to note that this approximation is based on our proposed theorems in which most of the intermediate layers do not require training.

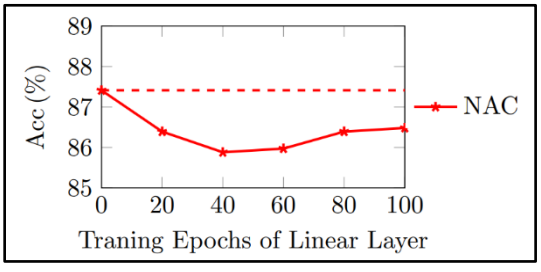

Figure 7: The effects of training final linear layer of NAC on the accuracy,

where x-axis denotes the training epochs of the final linear layer and y-axis denotes the averaged accuracy of acquired architecture α using the corresponding weights.

VI. FUTURE SCOPE & DIRECTIONS

Time series data, inherently characterized by unpredictable noise and uncertainty in the data-generating process, necessitates robust models capable of quantifying uncertainty for enhanced reliability and utility [36]. The incorporation of uncertainty quantification offers a probabilistic measure, contributing to the model's ability to express confidence in predictions and system state estimates. Uncertainty quantification provides a probabilistic measure of the confidence to the predictions made by the model and to the system state estimatesaiding in the understanding of the range and likelihood of potential outcomes [37]. This holds particular significance when Graph Neural Networks (GNNs) are employed in decision-making processes within high-stakes domains, such as financial forecasting, healthcare monitoring [38], or traffic prediction in smart cities [39]. Despite advancements, a gap persists in existing GNN models, predominantly providing point estimates [40], [41], thereby inadequately addressing potential uncertainties. This underscores a critical research direction: the development of sophisticated uncertainty quantification methods for GNNs to navigate the complexities inherent in time series data.Strategies such as pretraining, transfer learning, and the utilization of large models are emerging as potent approaches to enhance GNN performance in time series analysis, particularly in scenarios involving sparse or diverse data. These techniques rely on leveraging learned representations from one or more domains to improve performance in related domains. Noteworthy examples include Panagopoulos et al.'s model-agnostic meta-learning schema for predicting the spread of COVID-19 in data-limited cities and Shao et al.'s pre-training enhanced framework for spatial-temporal GNNs. The exploration of pre-training strategies and GNN transferability for time series tasks constitutes a burgeoning research area, particularly in the contemporary era of generative AI and large models, showcasing the potential for a unified, multimodal model to address diverse tasks.

CONCLUSION

This comprehensive survey addresses the knowledge gap in the realm of graph neural networks for time series analysis (GNN4TS) through an exhaustive review of recent advancements and the establishment of a unified taxonomy for categorizing existing works from both task- and methodology-oriented perspectives. Pioneering in its approach, the survey encompasses a diverse array of tasks, including forecasting, classification, anomaly detection, and imputation, providing an intricate understanding of the current state of the art in GNN4TS. The survey meticulously explores the complexities of spatial and temporal dependencies modeling and overall model architecture, offering a nuanced classification of individual studies. By emphasizing the expanding applications of GNN4TS across various sectors, the survey underscores its versatility and potential for future growth. This compilation serves as a valuable resource for machine learning practitioners and domain experts seeking insights into the latest advancements in this field. Lastly, the survey proposes potential future research directions, presenting insights aimed at guiding and inspiring subsequent work in GNN4TS.Additionally, the survey introduces the first linear complexity Neural Architecture Search (NAS) algorithm for Graph Neural Networks (GNNs), termed Neural Architecture Coding (NAC). NAC, solved through sparse coding, features a no-update scheme for model weights in GNNs, capitalizing on the inherent linearity and orthogonality of model weights within GNNs. Extensive experiments affirm that NAC achieves superior accuracy (up to 19.9%) and significantly faster convergence (up to 200×) compared to state-of-the-art NAS-GNN baselines. Several promising directions for future exploration are identified, including the investigation of deep neural networks adhering to the mild conditions of NAC for expanded applicability. Further exploration into the efficiency of different sub gradient methods for solving the sparse coding objective is recommended. Additionally, a proposed avenue for investigation involves jointly learning the search space and architectural representation to enhance the expressive ability of searched architectures.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” IEEE TNNLS, vol. 32, no. 1, pp. 4–24, 2020.

[2] Z. A. Sahili and M. Awad, “Spatio-temporal graph neural networks: A survey,” arXiv preprint, vol. abs/2301.10569, 2023.

[3] W. Jiang and J. Luo, “Graph neural network for traffic forecasting: A survey,” Expert Syst. Appl., p. 117921, 2022.

[4] X. Zhang, M. Zeman, T. Tsiligkaridis, and M. Zitnik, “Graph-guided network for irregularly sampled multivariate time series,” in ICLR, 2022.

[5] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang, “Multivariate time-series anomaly detection via graph attention network,” in ICDM, 2020, pp. 841– 850.

[6] A. Cini, I. Marisca, and C. Alippi, “Filling the g ap s: Multivariate time series imputation by graph neural networks,” in ICLR, 2022.

[7] Q. Wen, L. Yang, T. Zhou, and L. Sun, “Robust time series analysis and applications: An industrial perspective,” in KDD, 2022, pp. 4836–4837.

[8] B. Lim and S. Zohren, “Time-series forecasting with deep learning: a survey,” Philos. Trans. Royal Soc. A PHILOS T R SOC A, vol. 379, no. 2194, p. 20200209, 2021.

[9] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, “Deep learning for time series classification: a review,” DMKD, vol. 33, no. 4, pp. 917–963, 2019.

[10] A. Blazquez-Garc ´ ´ıa, A. Conde, U. Mori, and J. A. Lozano, “A review on outlier/anomaly detection in time series data,” ACM Computing Surveys, vol. 54, no. 3, pp. 1–33, 2021.

[11] C. Fang and C. Wang, “Time series data imputation: A survey on deep learning approaches,” arXiv preprint, vol. abs/2011.11347, 2020.

[12] A. A. Cook, G. Mısırlı, and Z. Fan, “Anomaly detection for iot time-series data: A survey,”

IEEE IoT Journal, vol. 7, no. 7, pp. 6481–6494, 2019.

[13] S. Wang, J. Cao, and S. Y. Philip, "Deep learning for spatiotemporal data mining: A survey," IEEE TKDE, vol. 34, no. 8, pp. 3681–3700, 2020.

[14] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," IEEE TITS, vol. 23, no. 5, pp. 3904–3924, 2020.

[15] G. Jin, Y. Liang, Y. Fang, J. Huang, J. Zhang, and Y. Zheng, "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey," arXiv preprint, vol. abs/2303.14483, 2023.

[16] S. Rahmani, A. Baghbani, N. Bouguila, and Z. Patterson, "Graph neural networks for intelligent transportation systems: A survey," IEEE TITS, 2023.

[17] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstr ¨ om, "Detecting spacecraft anomalies using lstms and ¨ nonparametric dynamic thresholding," in KDD, 2018, pp. 387– 395.

[18] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in KDD, 2019, pp. 2828–2837.

[19] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in ICLR, 2022.

[20] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, "Learning graph structures with transformer for multivariate time-series anomaly detection in iot," IEEE IoT Journal, vol. 9, no. 12, pp. 9179–9189, 2021.

[21] S. Moritz and T. Bartz-Beielstein, "imputets: time series missing value imputation in r." R J., vol. 9, no. 1, p. 207, 2017.

[22] M. Saad, M. Chaudhary, F. Karray, and V. Gaudet, "Machine learning based approaches for imputation in time series data and their impact on forecasting," in SMC, 2020, pp. 2621–2627.

[23] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," Scientific reports, vol. 8, no. 1, p. 6085, 2018.

[24] M. Jin, Y. Zheng, Y.-F. Li, S. Chen, B. Yang, and S. Pan, "Multivariate time series forecasting with dynamic graph neural odes," IEEE TKDE, 2022.

[25] G. Lai, W. Chang, Y. Yang, and H. Liu, "Modeling long- and shortterm temporal patterns with deep neural networks," in SIGIR, 2018, pp. 95–104.

[26] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," Machine Learning, vol. 108, pp. 1421–1441, 2019

[27] A. D. Richardson, M. Aubinet, A. G. Barr, D. Y. Hollinger, A. Ibrom, G. Lasslop, and M. Reichstein, "Uncertainty quantification," Eddy covariance: A practical guide to measurement and data analysis, pp. 173–209, 2012.

[28] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," Information Fusion, vol. 76, pp. 243–297, 2021.

[29] Y. Li, B. Qian, X. Zhang, and H. Liu, "Knowledge guided diagnosis prediction via graph spatial-temporal network," in SDM, 2020, pp. 19–27.

[30] H. Wen, Y. Lin, X. Mao, F. Wu, Y. Zhao, H. Wang, J. Zheng, L. Wu, H. Hu, and H. Wan, "Graph2route: A dynamic spatial-temporal graph neural network for pick-up and delivery route prediction," in KDD, 2022, pp. 4143–4152.

[31] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in AAAI, 2019, pp. 922–929.

[32] D. Huang, J. Bartel, and J. Palowitch, "Recurrent graph neural networks for rumor detection in online forums," arXiv preprint, vol. abs/2108.03548, 2021.

[33] L. Wu, P. Cui, J. Pei, L. Zhao, and L. Song, Graph neural networks. Springer, 2022.

[34] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in NeurIPS, 2017, pp. 1024–1034.

[35] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," in ICML, vol. 80, 2018, pp. 941–949.

[36] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," in ICLR, 2018.

[37] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. Hsieh, "Clustergcn: An efficient algorithm for training deep and large graph convolutional networks," in KDD, 2019, pp. 257–266.

[38] X. Wang, D. Wang, L. Chen, and Y. Lin, "Building transportation foundation model via generative graph transformer," arXiv preprint, vol. abs/2305.14826, 2023.

[39] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," arXiv preprint, vol. abs/2306.08302, 2023.

[40] G. Panagopoulos, G. Nikolentzos, and M. Vazirgiannis, "Transfer graph neural networks for pandemic forecasting," in AAAI, 2021, pp. 4838–4845.

[41] X. Wang, G. Chen, G. Qian, P. Gao, X.-Y. Wei, Y. Wang, Y. Tian, and W. Gao, "Large-scale multi-modal pre-trained models: A comprehensive survey," arXiv preprint, vol. abs/2302.10035, 2023.

[42] W. Xi, A. Jain, L. Zhang, and J. Lin, "Lb-simtsc: An efficient similarity-aware graph neural network for semi-supervised time series classification," arXiv preprint, vol. abs/2301.04838, 2023.

[43] D. Zha, K.-H. Lai, K. Zhou, and X. Hu, "Towards similarity-aware time-series classification," in SDM, 2022, pp. 199–207.

[44] K. Chen, M. Feng, and T. S. Wirjanto, "Multivariate time series anomaly detection via dynamic graph forecasting," arXiv preprint, vol. abs/2302.02051, 2023.

[45] Y. Zheng, H. Y. Koh, M. Jin, L. Chi, K. T. Phan, S. Pan, Y.-P. P. Chen, and W. Xiang, "Correlation-aware spatial-temporal graph learning for multivariate time-series anomaly detection," arXiv preprint, vol. abs/2307.08390, 2023.

[46] W. Zhang, C. Zhang, and F. Tsung, "Grelen: Multivariate time series anomaly detection from the perspective of graph relational learning," in IJCAI, 2022, pp. 2390–2397.

[47] G. Shen, W. Zhou, W. Zhang, N. Liu, Z. Liu, and X. Kong, "Bidirectional spatial-temporal traffic data imputation via graph attention recurrent neural network," Neurocomputing, vol. 531, pp. 151–162, 2023.

[48] X. Kong, W. Zhou, G. Shen, W. Zhang, N. Liu, and Y. Yang, "Dynamic graph convolutional recurrent imputation network for spatiotemporal traffic missing data," KBS, vol. 261, p. 110188, 2023.

[49] M. Jin, G. Shi, Y.-F. Li, Q. Wen, B. Xiong, T. Zhou, and S. Pan, "How expressive are spectral-temporal graph neural networks for time series forecasting?" arXiv preprint, vol. abs/2305.06587, 2023.

[50] A. Cini*, I. Marisca*, F. Bianchi, and C. Alippi, "Scalable spatiotemporal graph neural networks," in AAAI, 2023.

[51] H. Yu, T. Li, W. Yu, J. Li, Y. Huang, L. Wang, and A. Liu, "Regularized graph structure learning with semantic knowledge for multi-variates time-series forecasting," in IJCAI, 2022, pp. 2362–2368.