Temporal Graph Neural Networks

Erik Söhner, Max Helm*, Benedikt Jaeger*
*Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
Email: e.soehner@tum.de, helm@net.in.tum.de, jaeger@net.in.tum.de

Abstract—**Graph Neural Networks (GNNs) have become a fundamental tool for working with graph-structured data. They have shown high accuracy in making predictions in a wide range of applications and are now playing an important role in the field of machine learning. To combine the graph structure with temporal data, the Temporal Graph Neural Network emerged, which brings improved predictive performance to a variety of tasks. The paper shows common patterns in temporal and traditional graph neural network architectures and provides information on applications and open challenges.**

Index Terms—**Temporal Graph Neural Networks**

1. Introduction

Neural networks play an important role in machine learning. These models help machine learning make accurate and efficient predictions [1]. Their ability to recognize patterns allows them to process data where traditional machine learning algorithms struggle due to the complexity of the data. As a result, many types of neural networks are used in a wide range of real-world applications. These include image and speech recognition, natural language processing, autonomous vehicles, and personalized recommendation systems [1]. For graph-structured data, graph neural networks have emerged, contrary to the traditional neural networks that are prevalent for grid-like data. As neural networks show promise for more complex tasks, new neural network architectures are rapidly being developed. Temporal Graph Neural Networks (TGNN) are one of the latest NN architectures. They extend the graph neural network with additional modeling of temporal dependencies on spatial features. In this paper, we give some background on machine learning algorithms, explain graph neural network architectures, and show how temporal graph neural networks extend static GNN architectures. We discuss the possibilities of TGNNs with new frameworks such as PyTorch Geometric Temporal, as well as future challenges in this area of research. The paper also presents a TGNN architecture used for Internet traffic prediction, as well as an overview of other TGNN applications.

2. Background on Machine Learning Algorithms

The most basic type of a Neural Network is the Multi Layer Perceptron (MLP). It only consists of fully-connected layers, namely an input layer, an output layer and hidden layers in between. All nodes of a layer have weighted edges, the learnable parameters, to all nodes of their neighbor layers. The information of a node becomes the weighted average of the node information from the previous layer put in an activation function. This is known as message passing. [2]

Another popular type is Convolutional Neural Network (CNN). They consist of a number of convolutional layers and a fully connected layer to generate the output. As input, it takes a multidimensional array, typically an image with its dimensions: height, width, color channels. The convolution layer computes the convolution operation on the layer's input. This operation first applies filters containing the learnable parameters, a matrix multiplication, then it applies an activation function. Between the layers, the sizes and number of dimension can change, for the fully connected layer, it is arranged in single-dimension array. [3]

2.1. Training of Neural Networks

The training of neural networks is an iterative process in which the output of the neural network is used to make a meaningful change in the learnable parameters of the network. First, a cost function measures the discrepancy between the network's actual output and its optimal output. The optimal output can either be known or the plausibility of the actual output can be checked, depending on the learning method of the neural network. Then, a vector of optimal changes for all learnable parameters is computed to minimize the cost function, called the gradient, and applied to the parameters. [4]

3. Graph Neural Networks

Traditional neural networks perform well on a wide range of tasks, but leave a lot of potential when working with graph-structured data. Therefore, GNNs are designed to exploit the existing dependencies. Consequently, GNNs have shown superior performance in a number of domains where they outperform other machine learning (ML) algorithms. Examples of domains where GNNs are successfully used are social network analysis, drug discovery, or recommendation systems. [5]

The architecture of GNNs can be broadly classified into spatial and spectral GNNs [6]. Architectures of GNNs are highly variant and difficult to generalize. The following descriptions of architectures can be understood as feasible architectures that use common functions and patterns across various GNN architectures.

3.1. Spatial Graph Neural Networks Architecture

In spatial GNNs, each layer updates the graph representation. Here we assume a traditional GNN with a static graph, so the only thing that changes in each layer are the node features. The message passing of a hidden layer is processed for each node individually. The neighbor node features are the input for a message-passing function, which can be decomposed into four functions that may occur. [6]

1) **Transformation function** is a matrix multiplication on each input's node feature map. This is where attention mechanisms, a multiplication by the corresponding edge's weight, are usually located. In general, this weight matrix can contain learnable or static parameters or edge features.
2) **Aggregation function** takes the same feature from all transformed node feature maps. The aggregation is applied to each feature, creating a single feature map. Typical aggregation functions are min, max, sum, or average.
3) **Update function** combines the remaining features from previous operations with the node features in element-wise operations.
4) **Activation function** amplifies the node features. Typical activation functions include sigmoid, ReLU, and softmax.

3.2. Spectral Graph Neural Networks Architecture

A major difference to spatial GNNs is the global nature of message passing in the spectral domain. In order to operate in the spectral domain, the feature matrix must first be transformed by multiplying it by the eigenvector of the graph's Laplacian matrix. Then, the functions present in a layer can be applied. [6]

1) **Convolution** is the multiplication of the transformed matrix with a diagonal matrix of weights
2) **Activation function**
3) **Pooling** takes a single value from a neighborhood of nodes, reducing the dimension of the feature map. Typical pooling functions are max pooling, average pooling, attention pooling. For node classification tasks, the output layer of the spectral GNN is often a fully connected layer.

4. Temporal Graph Neural Networks

Traditional Graph Neural Networks are limited in applications that work with time-varying data. These GNN models have difficulty modeling temporal dynamics because the graph structure they work with is static, neglecting the changing relationships that actually exist. For example, in social network analysis, a model must understand the changing social interactions over time in order to predict influential individuals or discover communities. Similarly, in financial applications like risk assessment of investment decisions, market dynamics must be taken into account to make accurate predictions. To overcome these limitations, Temporal Graph Neural Networks (TGNNs) have become a promising extension to traditional GNNs and a relevant area of research. The models allow the modeling of temporal dependencies in addition to the spatially arranged data. [7]

4.1. Temporal Graph Neural Network Architectures

In the past, researchers have proposed a number of TGNN architectures. In addition to the algorithmic classification of GNNs, namely spatial or spectral, there is another major design decision for TGNNs in the time variant method. The temporal information can be directly defined in the graph structure, or the GNN can be extended by operations handling the temporal information. A common approach are Hybrid Graph Neural Networks, where time-varying information is processed in another ML algorithm. [7]

Temporal information can be a static information: a time stamp, which can be modeled as a node feature, or the temporal distance between nodes, which can be modeled as an edge feature. An example for this is pandemic forecasting [8].

Temporal information related to a node or an edge can also be dynamic, it can be referred to as time varying information, e.g. time series information. In this case, GNN does not need to be extended yet, because the information can be modeled as multidimensional features. Time series information can also be passed as an argument into the message passing function, where GNN layers correspond to specific time steps [9]. In spectral GNNs, this kind of information can be used to set the parameters for convolution kernels [10].

If the graph is dynamic, i.e. links change over time, an extension of the GNN is inevitable. A dynamic graph representation can be a sequence of graph snapshots over time, where the individual graphs serve as input for a Graph Attention Network (GAT), to be later merged in an output function [11]. Another option is the temporal evolution of the graph, where in each layer the neighbor nodes are defined by the corresponding graph snapshot of the layer at a given time [12].

Hybrid GNNs for processing temporal information can be used, too. There is no standard way for information flow between time and graph modules. Nevertheless, the capabilities of specific architectures can be used to take advantage of the different ways of handling time. [7]

- 1D-CNN time module: A convolutional neural network with one-dimensional input is used to extract temporal patterns. The convolution operation allows the model to have a small number of parameters as well as translation invariance, so that patterns are detected regardless of their temporal position. In addition, manual modification can help the model to better fit the data. 1D-CNN can complement the GNN in a way that can be considered as sandwiched, where its input and output is the output and input for a GNN layer, respectively. [13]
- LSTM time module: Long Short-Term Memory is a type of Recurrent Neural Network (RNN). It allows the network to selectively remember

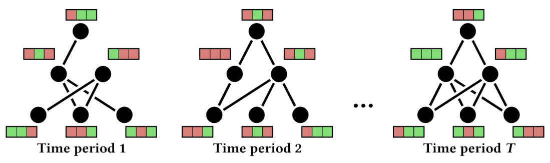
and forget information based on its relevance to the context. The component can model sequential patterns, such as trends, and seasonality. An application here is to predict PV power forecasting by modeling spatial and temporal dependencies between power plants. [14]

4.2. PyTorch Geometric Temporal

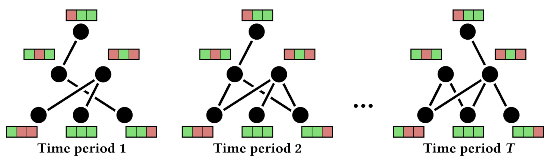
As machine learning algorithms evolve in research and real-world applications, software emerges that enables widespread use and rapid implementation of machine learning models. However, in the early research stages of new architectures, well-suited libraries often do not exist and implementation tasks are complicated. Motivated to create an open source machine learning algorithm that could handle non-static node features in graphs, Rozemberczki *et al.* came up with the PyTorch Geometric Temporal framework. [15]

Their goal was to create a user-friendly and functional software. For easy inspection of the runtime state, they designed the software in a modular way with limited number of public methods. Furthermore, the system provides test coverage, documentation, practical tutorials, continuous integration, package indexing, and frequent releases.

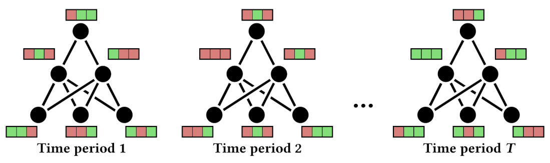
Providing this framework, the authors claim that PyTorch Geometric Temporal is the first deep learning library designed for neural spatiotemporal signal processing. Figure 1 shows the different scenarios of GNN's non-static properties that the framework is able to handle.



(a) Dynamic graph with temporal signal.



(b) Dynamic graph with static signal.



(c) Static graph with temporal signal.

Figure 1: PyTorch scenarios [15]

Examining the framework's predictive performance, the authors find that PyTorch Geometric Temporal has similar predictive performance to recurrent neural networks on regression tasks. Among possible future directions, they mention considering continuous time or time differences between temporal snapshots that are not constant. Another possibility they see is the inclusion of temporal models that operate on curved spaces, like hyperbolic or spherical spaces.

4.3. Current Challenges in Temporal Graph Neural Network Development

While the potential of TGNNs is undeniable, there are still challenges that may slow down TGNN research.

A prominent problem is the lack of benchmarking capabilities. Models are trained for a specific problem, but due to limited standardized datasets and evaluation metrics, the models are not tested on datasets from different domains. This makes it uncertain whether they are suitable for other applications. [7]

A further difficulty for the rather new TGNNs is the need to adapt the learning methods to avoid oversmoothing. For GNNs, techniques such as dropout, virtual nodes, and neighbor sampling exist. Due to the variety and complexity of architectures, no solution can serve as a general solution [16]. Therefore, a lot of experiments need to be done on the even more complex TGNNs.

Another well-known problem is that institutions working with privacy-critical data are often unable to publish data. This is related to missing methods to create privacy-preserving representations. One solution is federated learning, a training method for dealing with data isolation between different sources. Guannan Lou *et al.* have demonstrated the effectiveness of a federated learning framework for a TGNN. [17]

5. Related Work

There is a range of applications where TGNNs show good performance.

Traffic prediction is a major domain for TGNN research. The temporal GNNs have superior performance over traditional GNNs in tasks like traffic planning and route planning. Their predictions show to be more accurate and better handle dynamic traffic conditions, such as traffic fluctuations and congestion. [18]

Pandemic forecasting for the TGNN became a big research domain during covid pandemic. It has shown to achieve state-of-art forecasting. [8]

PV production forecasting is of great importance for the transition to renewable energy. TGNNs have shown to outperform state-of-the-art methods for PV forecasting [14]. Therefore, TGNNs have great potential for long-term applications such as infrastructure planning as well as short-term applications such as efficient dispatching of other sources or informed decisions related to energy markets.

5.1. Internet Traffic Forecasting using Temporal-Topological Graph Convolutional Networks

A prospective application for TGNNs can be internet traffic forecasting. Being part of everydays life, it is important the internet works on fast and reliable infrastructure. With billions of connected devices and exponentially growing traffic it needs good solutions to master this challenge. However, it opens up opportunities for continuous innovation in network technology and supporting algorithms. One advancement for infrastructure planning and network resource management lies in accurate internet traffic prediction.

Internet traffic prediction belongs to the class of time series forecasting problems. In this field linear prediction methods are used as well as neural networks, being capable to model non-linear data. However existing neural network algorithms in internet traffic forecasting often ignore network topology as they mainly model temporal data of traffic flow series. [19]

Zhenjie Yao *et al.* [19] proposed "Temporal-Topological Graph Convolutional Networks" (TTGCN), a TGNN architecture modeling the links' throughput of internet traffic in time series with also capturing network topology for predicting internet traffic. The graph representation looks as follows: A network link makes a node in the TTGCN graph, where edges exist, when the links are connected to the same router. The model processes in turn the temporal convolution and graph convolution, as illustrated in Figure 2. The temporal convolution extracts the temporal features while the graph convolution uses the new representation connecting it to the topological information. The temporal convolution is a gated linear unit that takes as input the feature matrix of the graph, initially the time series data of the links. Two matrices are generated by multiplying the input by the two different convolution kernels set in the current layer. A sigmoid function is applied to one of the matrices, which is then merged with the other matrix by computing the Hadamard product. The output is an updated graph feature matrix with one feature less. The new feature matrix is passed to the graph convolution layer that works with the graph's spectral domain. Here the researchers came up with two different approaches for representing the graphs adjacency matrix. One is a normal adjacency matrix for the graph as described before, the other is defined

$$\hat{A}_{i,j} = \begin{cases} B, & \text{if link } i \text{ heads to tail of link } j \\ -1, & \text{if link } i \text{ heads to tail of link } j \text{ and v.v.} \\ -1, & \text{if link } i \text{ heads to head of link } j \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

where the optimal parameter B is to be found by testing for the smallest error. Head and tail are the routers that a directed link passes traffic to and from respectively. The main operation here is the multiplication of its input with the graphs's Laplacian matrix and a matrix of learnable parameters. Following this structure, the final temporal convolution layer's output contains one remaining node feature. At this stage the feature matrix is passed to a fully connected layer, whose results are the predictions for every network link. Obtaining only one time step's predictions, the later time steps' predictions need to be obtained recursively.

The model's performance was tested with data from the UKERNA academic network backbone by Simple Network Management Protocol (SNMP). Traffic of 18 links connecting 8 core routers was observed for over a month. Samples for all links were taken every 10 minutes. In the paper the MAE and RMSE are compared for different prediction models, namely Historical Average, ARIMA, a popular linear model for time-series forecasting, Gated Recurrent Unit, Spatio-Temporal Graph Convolution Networks (STGCN), a TGNN showing good performance in road traffic prediction. Both the temporal GNNs were measured with the normal and the advanced adjecency matrix: TTGCN+, STGCN+. In the test the model should predict the next 9 time steps after being initialized with data of 12 time steps. The models were trained with data of 31 days and were tested on the data of the 8 remaining days. The test results show best performance for TTGCN+ then STGCN+, TTGCN, STGCN, pointing out the proposed model achieved the best prediction performance. The paper does not provide information if the best adjacency matrix parameter was calculated for STGCN+, too. With TTGCN+ having a 13.7% lower RMSE than STGCN+ and over 10% lower RMSE than TTGCN+ with next higher parameter shows that having a good data representation is crucial for exploiting a prediction model's potential.

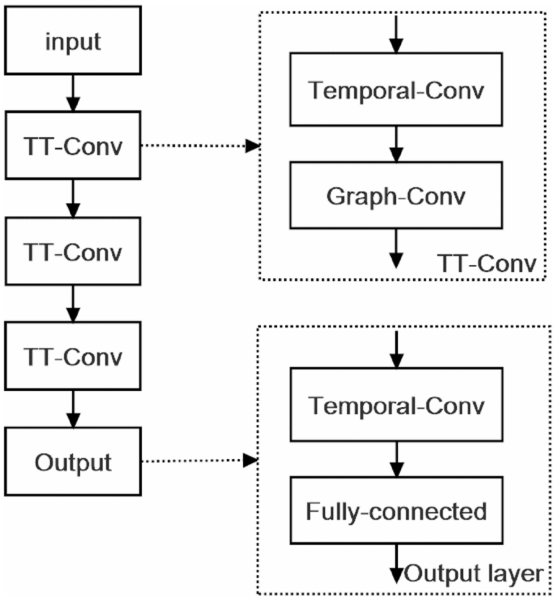


Figure 2: Architecture of TTGCN [19] ©2021 IEEE

6. Conclusion

In this paper, Temporal Graph Neural Networks are explained. It provides relevant information on machine learning algorithms and describes how common spectral and spatial GNNs are constructed. It can be concluded that TGNNs show good prediction performance on data with spatial and temporal relationships. Their architectures are usually combinations of common patterns in machine learning. One can expect that new TGNN architectures will emerge to leverage their capabilities in even more applications. However, there are challenges that need to be addressed. Contributions to benchmarking tools are needed, GNN methods against oversmoothing need to be adopted, further advances for federated learning must be driven.

References

[1] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.

[2] H. Taud and J. Mas, *Multilayer Perceptron (MLP)*. Cham: Springer International Publishing, 2018, pp. 451–455. [Online]. Available: https://doi.org/10.1007/978-3-319-60801-3_27

[3] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," 2015.

[4] F. Günther and S. Fritsch, "Neuralnet: training of neural networks." *R J.*, vol. 2, no. 1, p. 30, 2010.

[5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[6] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[7] Z. A. Sahili and M. Awad, "Spatio-Temporal Graph Neural Networks: A Survey," 2023.

[8] A. Kapoor, X. Ben, L. Liu, B. Perozzi, M. Barnes, M. Blais, and S. O'Banion, "Examining COVID-19 Forecasting using Spatio-Temporal Graph Neural Networks," 2020.

[9] L. Wang, A. Adiga, J. Chen, A. Sadilek, S. Venkatramanan, and M. Marathe, "CausalGNN: Causal-Based Graph Neural Networks for Spatio-Temporal Epidemic Forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, pp. 12 191–12 199, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/21479

[10] S. Hadou, C. I. Kanatsoulis, and A. Ribeiro, "Space-Time Graph Neural Networks," 2022.

[11] A. Fathy and K. Li, "Temporalgat: Attention-based dynamic graph representation learning," in *Advances in Knowledge Discovery and Data Mining*, H. W. Lauw, R. C.-W. Wong, A. Ntoulas, E.-P. Lim, S.-K. Ng, and S. J. Pan, Eds. Cham: Springer International Publishing, 2020, pp. 413–423.

[12] Y. Fan, M. Ju, C. Zhang, and Y. Ye, *Heterogeneous Temporal Graph Neural Network*, pp. 657–665. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611977172.74

[13] A. M. Karimi, Y. Wu, M. Koyuturk, and R. H. French, "Spatiotemporal Graph Neural Network for Performance Prediction of Photovoltaic Power Systems," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 17, pp. 15 323–15 330, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17799

[14] J. Simeunović, B. Schubnel, P.-J. Alet, and R. E. Carrillo, "Spatio-Temporal Graph Neural Networks for Multi-Site PV Power Forecasting," *IEEE Transactions on Sustainable Energy*, vol. 13, no. 2, pp. 1210–1220, 2022.

[15] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, A. Riedel, M. Astefanoaei, O. Kiss, F. Beres, G. López, N. Collignon, and R. Sarkar, "PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models," 2021.

[16] A. Longa, V. Lachi, G. Santin, M. Bianchini, B. Lepri, P. Lio, F. Scarselli, and A. Passerini, "Graph Neural Networks for temporal graphs: State of the art, open challenges, and opportunities," 2023.

[17] G. Lou, Y. Liu, T. Zhang, and X. Zheng, "STFL: A Temporal-Spatial Federated Learning Framework for Graph Neural Networks," 2022.

[18] Y. Li, W. Zhao, and H. Fan, "A Spatio-Temporal Graph Neural Network Approach for Traffic Flow Prediction," *Mathematics*, vol. 10, no. 10, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/10/1754

[19] Z. Yao, Q. Xu, Y. Chen, Y. Tu, H. Zhang, and Y. Chen, "Internet Traffic Forecasting using Temporal-Topological Graph Convolutional Networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.