Multivariate Time Series Forecasting with Graph Neural Networks - application to EEG

Tom Mariani tom.mariani68@gmail.com
Hugo Pavy hugo.pavy@etu.minesparis.psl.eu

January 4, 2026

1 Introduction and contributions

The goal of this work is to study the use of Graph Neural Networks (GNNs) for multivariate time series forecasting and apply this approach to EEG data. We build upon the architecture proposed in the paper *Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks* by Wu et al. (2020). Crucially, this framework implements a model that simultaneously learns the underlying graph relations and performs the forecasting task. The proposed model was evaluated on traffic, electricity, solar, and exchange rate datasets. Our main contribution is to provide result visualizations and apply this architecture to EEG data. Our implementation is available online at: GitHub repository.

Why EEG? EEG data extracted from the human brain is inherently a multivariate time series. Multiple sensors are placed on the patient's scalp to monitor electrical brain activity. While an intuitive graph structure exists based on the physical distance between sensors, it is not necessarily optimal for forecasting. The adopted approach allows us to move beyond this fixed spatial topology, determining whether the physical distance is the best predictor or if a learned graph better captures the complex functional dependencies.

Work Distribution Between the Two Students We did not assign specific tasks to each team member. Instead, we first collaborated to understand the paper. We then implemented the model from scratch in Python improving the code found in the paper's repository with a modular object-oriented architecture. Whenever one of us had time to contribute, he created a branch on Github, developed a new feature, and submitted a pull request for review by the other. This process ensured continuous code quality and knowledge sharing.

Use of Existing or New Experiments — Improvements Over the Original Method To deepen our understanding of the model, we decided to implement it entirely from scratch, initially relying only on the paper's descriptions. To validate our implementation and ensure we did not overlook any details, we later referred to the authors' original code. After successfully reproducing the paper's results, we applied the model to EEG data, including a dedicated preprocessing and analysis phase.

2 Method

We consider a multivariate time series $\mathbf{X} \in \mathbb{R}^{N \times T}$, where N is the number of variables (or sensors) and T is the length of the sequence. Our goal is to predict the future value of each sensor at a specific horizon h. For single-step forecasting, if $h = 1$, we predict the immediate next timestamp; if $h = 20$, we predict the value 20 steps into the future. The model is typically trained specifically for a given horizon h.

The model assumes the existence of a latent graph structure among the variables, meaning the time series are correlated. However, the exact dependency structure is unknown a priori. Therefore, the model aims to learn this structure automatically.

The learned graph is then used to aggregate information across sensors to facilitate prediction. The core architecture consists of a stack of layers, where each layer contains two main modules: a Time Convolution Module and a Graph Convolution Module.

The Graph Learning Layer To capture the hidden dependencies, the model learns an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ end-to-end via backpropagation. Unlike predefined graphs, this approach computes the dependencies dynamically based on the data. The graph learning layer typically uses learnable node embeddings to compute the similarity or influence between pairs of sensors, resulting in an adaptive adjacency matrix $\mathbf{A}$ that guides the message passing in the graph convolution module.

The Time Convolution Module For each sensor, we aim to extract meaningful temporal patterns, such as short-term trends and long-term periodicities (seasonality). To achieve this, the Time Convolution Module employs a *Dilated Inception Layer*.

- **Inception**: It applies standard 1D convolutions with multiple filter sizes (e.g., $1 \times 2, 1 \times 3, 1 \times 6, 1 \times 7$) in parallel. Smaller kernels capture local fluctuations, while larger kernels capture broader trends.
- **Dilation**: To capture long-term dependencies without drastically increasing computational cost, the convolutions are dilated. The dilation factor increases exponentially with the depth of the network, exponentially expanding the receptive field.

The output of this module results in C channels, where each channel represents a learned temporal embedding of the sensor at a specific time step.

The Graph Convolution Module After the Time Convolution Module augments the time series with high-level temporal features, the Graph Convolution Module fuses this information across the spatial dimension. It utilizes the learned adjacency matrix $\mathbf{A}$ to propagate information.

This module employs a *Mix-Hop Propagation* mechanism to allow information to flow not just between immediate neighbors, but also across multi-hop neighborhoods. Crucially, this operation occurs at each time step independently: a node aggregates information from its neighbors' features *at the same time step* and *within the same channel*. This spatial fusion allows the model to leverage the correlations between different sensors to improve the prediction accuracy for each individual node.

The Output Module Finally, the Output Module aggregates the latent representations learned at different depths via skip connections. These aggregated features are fed into a output layer, typically consisting of two 1×1 convolution layers (acting as a position-wise MLP). This module projects the high-dimensional channel features to the desired output dimension, generating the final prediction at horizon h for each sensor independently.

3 Data

3.1 EEG Data Characteristics

Electroencephalography (EEG) signals reflect a mixture of neural activity and non-neural noise sources, resulting in data that are highly correlated, non-stationary, and characterized by a low signal-to-noise ratio. A minimal yet principled preprocessing pipeline is therefore required to ensure numerical stability and learning efficiency, while avoiding strong assumptions that could bias downstream modeling.

Bandpass Filtering. All channels were bandpass-filtered with a high-pass cutoff at 0.5 Hz and a low-pass cutoff at 50 Hz. The high-pass filter attenuates slow baseline drifts caused by non-neural factors, while the low-pass filter removes power-line contamination (50 or 60 Hz) and restricts the signal to relevant frequency bands to measure neural activity based on experts knowledge.

Resampling. Following low-pass filtering, signals were downsampled from 512 Hz to 100 Hz. This sampling rate satisfies the Nyquist criterion for frequencies up to 50 Hz and substantially reduces sequence length and computational cost, which is critical for GNN-based forecasting models operating on long temporal windows.

Normalization. Channel-wise z-score normalization was applied to compensate for amplitude variability across electrodes and recording sessions. This ensures comparable numerical scales across nodes in the graph and improves optimization stability.

Optional Preprocessing. Common EEG preprocessing steps such as artifact removal, bad-channel rejection, or re-referencing were not applied. These procedures often rely on heuristic or manual decisions and may alter the spatial structure of the sensor graph. By limiting preprocessing to basic signal conditioning, the model is encouraged to learn robust representations directly from minimally processed data.

3.2 Discussion

The 0.5 Hz high-pass filter aligns with contemporary minimal preprocessing recommendations in EEG analysis (e.g., Delorme et al., 2023). Normalization is consistent with the preprocessing strategy of the original paper. The low-pass cutoff at 50 Hz represents a conservative compromise between noise suppression and information preservation. While physiologically motivated, this choice introduces an inductive bias whose impact should be acknowledged. Importantly, preprocessing does not impose assumptions on inter-channel relationships, which remain fully learned by the model. You can find in figure 2 the impact of the preprocessing on the data

4 Results

This section reports numerical results on real multivariate time series. We first reproduce benchmark experiments from the original MTGNN paper to validate our implementation, then extend the evaluation to EEG data, a novel application domain. Results are assessed using both forecasting metrics and analysis of the learned graph structures.

Reproduction on Benchmark Dataset To validate the correctness of our implementation, we reproduced the experiments of MTGNN on the Solar Energy dataset from the Multivariate Time Series Data Repository. This dataset consists of multiple correlated time series representing solar power production across different locations and is commonly used to benchmark spatio-temporal forecasting models.

We compare MTGNN against a classical autoregressive (AR) baseline across multiple forecasting horizons. Table 2 reports the forecasting errors for horizons of 3, 6, 12, and 24 time steps.

MTGNN consistently outperforms the AR baseline, particularly for longer horizons, confirming its ability to capture long-range temporal dependencies and inter-variable correlations. These results are in line with those reported in the original paper, validating our experimental setup. We used the same metrics: Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR).

Application to EEG Time Series We next evaluate MTGNN on multichannel EEG recordings, inside a single session run experiment for a The same evaluation protocol is applied, comparing MTGNN to an AR baseline across identical forecasting horizons. Results are reported in Table 1.

Table 1: Forecasting performance on EEG data.

Model	Horizon 3	Horizon 6	Horizon 12	Horizon 24
AR-RSE	0.912	0.980	0.956	0.938
MTGNN-RSE	0.852	1.034	0.976	0.958
AR-CORR	0.283	0.088	0.087	0.104
MTGNN-CORR	0.387	0.160	0.140	0.067

In terms of RSE, the AR model consistently outperforms MTGNN, raising questions about the practical benefit of such a large and complex model. However, MTGNN appears to capture the underlying trends more accurately, as reflected by significantly higher correlation scores. This difference is particularly evident when comparing the prediction plots to the ground-truth data, cf Figure 1.

Graph Learning Analysis Beyond quantitative performance, MTGNN provides access to a learned adjacency matrix that encodes inferred relationships between nodes. Figure 3 visualizes the learned graph projected onto the spatial layout of EEG electrodes.

Interestingly, the topology of the learned graph diverges significantly from a standard nearest-neighbor configuration. This indicates that long-range dependencies between spatially distant electrodes carry critical information, suggesting that functional connectivity in this context is not strictly bound by geometric proximity.

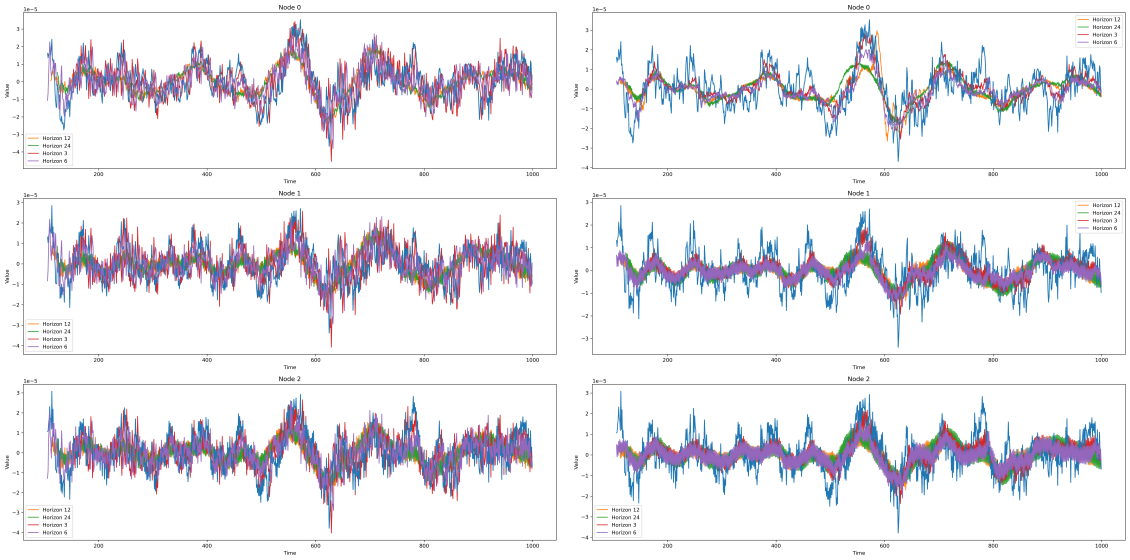

Figure 1: Comparison of prediction and true signal for 10 seconds signal and 3 electrodes, left AR model, right MTGNN

Discussion and Limitations

- A major limitation of this EEG application is that the results are based on a single run for one subject only. Moreover, due to its high computational cost, the model is not suitable for real-time applications.
- The train/validation/test split is performed chronologically. As a result, the test set is temporally distant from the training data. This gap of a few minutes may partly explain the relatively poor performance.
- The grid search on EEG data was limited to a small subset of hyperparameters (see Table 3); a more systematic exploration should be conducted.

Conclusion In conclusion, we successfully reproduced the MTGNN model and applied it to preprocessed EEG data, extending its use to a new application domain. We complemented the quantitative results with signal predictions and learned graph visualizations, providing insights into both temporal dynamics and inter-channel relationships. In addition, we introduced an autoregressive (AR) baseline, which serves as a simple yet informative point of comparison for evaluating the benefits and limitations of MTGNN.

5 Appendix

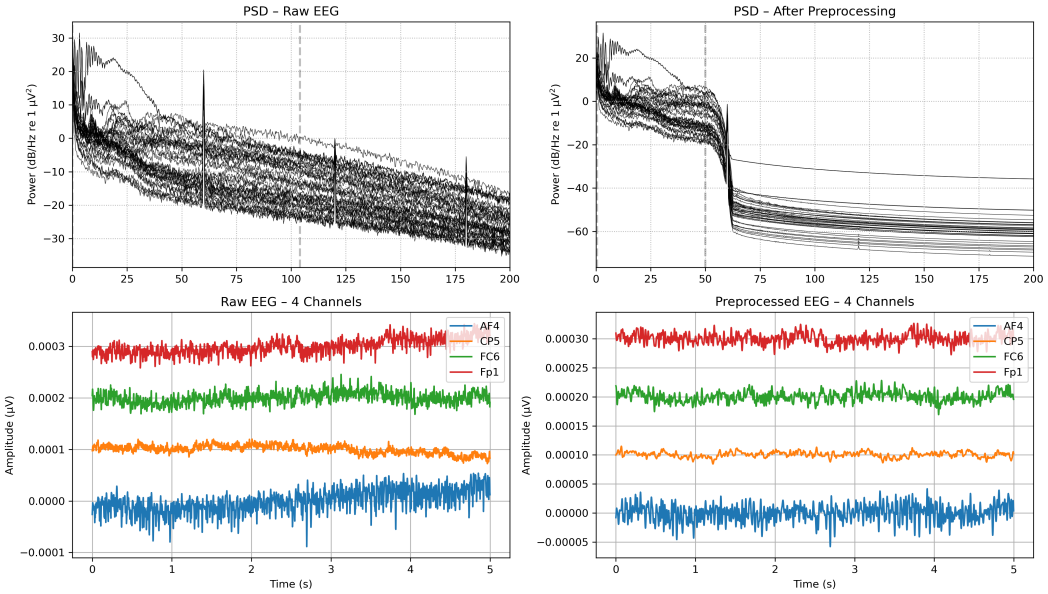

Figure 2: Example EEG channel before and after bandpass filtering, illustrating the removal of slow drifts and high-frequency noise.

Table 2: Forecasting performance on the Solar dataset.

Model	Horizon 3	Horizon 6	Horizon 12	Horizon 24
AR-RSE	0.259	0.341	0.446	0.546
MTGNN-RSE	0.215	0.263	0.330	0.410
AR-CORR	0.967	0.940	0.894	0.837
MTGNN-CORR	0.978	0.966	0.947	0.910

Table 3: Grid search for MTGNN at horizon 3 with EEG - changing the retain ratio β - result on validation data

β	0.05	0.25	0.5
RSE	0.603	0.5931	0.5935
CORR	0.0012	0.0012	0.0013

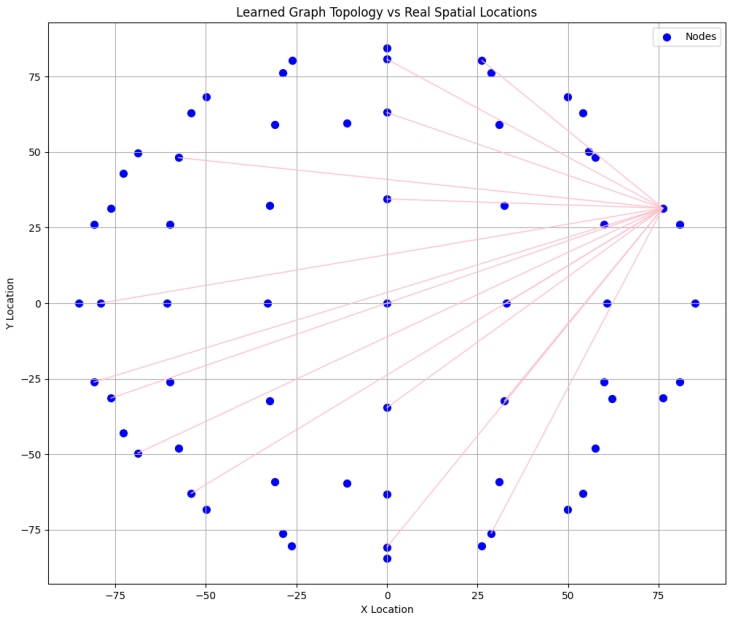

Figure 3: Graph structure learned by MTGNN, projected onto the EEG sensor layout. Example for one channel, one node.