International Journal of Data Science and Analytics (2023) 16:317–332
https://doi.org/10.1007/s41060-022-00349-6

REGULAR PAPER

Graph neural networks for multivariate time series regression with application to seismic data

Stefan Bloemheuvel[1,2] · Jurgen van den Hoogen[1,2] · Dario Jozinović[3,4,7] · Alberto Michelini[3] · Martin Atzmueller[5,6]

Received: 1 April 2022 / Accepted: 17 July 2022 / Published online: 30 August 2022

Abstract
Machine learning, with its advances in deep learning has shown great potential in analyzing time series. In many scenarios, however, additional information that can potentially improve the predictions is available. This is crucial for data that arise from e. g., sensor networks that contain information about sensor locations. Then, such spatial information can be exploited by modeling it via graph structures, along with the sequential (time series) information. Recent advances in adapting deep learning to graphs have shown potential in various tasks. However, these methods have not been adapted for time series tasks to a great extent. Most attempts have essentially consolidated around time series forecasting with small sequence lengths. Generally, these architectures are not well suited for regression or classification tasks where the value to be predicted is not strictly depending on the most recent values, but rather on the whole length of the time series. We propose TISER-GCN, a novel graph neural network architecture for processing, in particular, these long time series in a multivariate regression task. Our proposed model is tested on two seismic datasets containing earthquake waveforms, where the goal is to predict maximum intensity measurements of ground shaking at each seismic station. Our findings demonstrate promising results of our approach—with an average MSE reduction of 16.3%—compared to the best performing baselines. In addition, our approach matches the baseline scores by needing only half the input size. The results are discussed in depth with an additional ablation study.

Keywords Graph neural networks · Time series · Convolutional neural networks · Sensors · Regression · Earthquake ground motion · Seismic network

1 Introduction

In today's world, advances in hardware and wireless network technology have opened the path for energy-efficient, multifunctional and low-cost sensors [1]. Spread across a large geographical region, a set of sensors can then form a sensor network used for data collection and analysis [2], in particular considering large-scale time series data. Example domains where such real-world sensor data is analyzed

Stefan Bloemheuvel and Jurgen van den Hoogen have contributed equally to this work.

✉ Stefan Bloemheuvel
s.d.bloemheuvel@jads.nl

Jurgen van den Hoogen
j.o.d.hoogen@jads.nl

Dario Jozinović
djozinovi@gmail.com

Alberto Michelini
alberto.michelini@ingv.it

Martin Atzmueller
martin.atzmueller@uni-osnabrueck.de

1 Tilburg University, Tilburg, The Netherlands

2 Jheronimus Academy of Data Science, 's-Hertogenbosch, The Netherlands

3 Istituto Nazionale di Geofisica e Vulcanologia, Rome, Italy

4 Department of Science, Università degli Studi Roma Tre, Rome, Italy

5 Semantic Information Systems Group, Osnabrück University, Osnabrück, Germany

6 German Research Center for Artificial Intelligence (DFKI), Osnabrück, Germany

7 Present Address: Swiss Seismological Service (SED) at ETH Zurich, Zurich, Switzerland

include, e. g., traffic [3], weather [4] and seismology [5], in particular regarding time series regression and classification, e. g., [6,7].

Recently, there have been considerable advances in deep learning methods, in particular regarding CNNs, with respect to their ability to automatically find structure and meaningful features in the data. This leads to powerful (implicit) feature construction and computationally efficient models, e. g., [8,9] for time series. However, if only the time series data are examined, then some aspects of the sensor data are left unseen, i. e., the spatial relations of sensors in datasets that are geographically grounded.

Consequently, researchers have developed deep learning techniques to perform time series analysis like forecasting [10], anomaly detection [11] and imputation [12], with data arising from networks (i. e., graphs), called graph neural networks (now referred to as *GNNs*), which we also focus on in this paper. However, if the predicted value does not rely more on recent values from the input than early values (known as Time Series Extrinsic Regression (TSER) [7]), the aforementioned models are not adequate for the task.

Therefore, in this paper we tackle the problem of multivariate time series regression, for which we present a novel GNN-based architecture named TISER-GCN. Our evaluation applies high-frequency network-based seismic data demonstrating the efficacy of our proposed approach.

Previous attempts for tackling similar time series problems with graph-based methods have been made by [5,13,14], yet each has some shortcomings. van den Ende and Ampuero [5] mention that they designed a GNN for the localization of earthquakes from waveform data. However, they only append the (latitude, longitude) information to the time series being handled by a CNN. Therefore, while prediction scores improved, no actual GNN layers were used. Second, [13] proposed a graph partitioning algorithm that works together with a CNN. However, they make use of classical graph theory techniques and a GNN method is not applied. Lastly, [14] recently suggested a method that uses CNNs and GNNs for seismic event classification. However, (1) no spatial information is used at all, i. e., each edge has a weight of 1, nor (2) meta information about the stations is added, and (3) only three nodes are examined for each observation, which could be difficult to interpret as a full-fledged/complex network.

Therefore, we propose a larger scale GNN architecture that can process multivariate time series for such a regression task. By combining the capabilities of convolutional layers (feature extraction) and graph convolutional layers (spatial information), our model can manage the feature sizes that are common in high-frequency time series data arising from multiple sensors.

We test our proposed model on network-based seismic data, which serve as an intuitive domain where GNN models could be operated due to the naturally geographical grounded sensors. The model is inspired by the work presented in [15] and [16], which functions as our most prominent baseline. In their work, the maximum ground shaking at a set of seismic stations is predicted by tackling this as a regression problem. Their model used convolutional layers to extract useful features from a given time series. We start from their work as a departure point and illustrate how to design the deep learning model structure using GNNs for such a task.
Our contributions are summarized as follows:

1. We propose a method to perform multivariate regression on time series originating from graph-structured data. For this, we present an architecture utilizing convolutional and graph convolutional layers that is also adjustable for other use cases or datasets, e. g., time series classification tasks.
2. We evaluate our model thoroughly on two seismological datasets that differ significantly from one another evidencing the generality and potential of the proposed GNN-based architecture in this task. We discuss our results in detail and perform a comparison against several baseline models (in particular [15], but also [14,17]) and traditional machine learning methods.
3. Finally, we systematically analyze the capabilities of our model in detail by comprehensive experimentation adjusting several hyperparameters in our proposed workflow.

The article is further structured as follows: we discuss related work in Sect. 2, which provides the necessary background on deep learning, graphs and GNNs. Next, Sect. 3 introduces the dataset, our method and training settings. After that, Sect. 4 presents our results and discusses these in the context of a model-based comparison. Finally, Sect. 5 concludes with a summary and outlines interesting directions for future work.

2 Background and related work

This section briefly outlines the background and related work on graphs and deep learning in general, CNNs, GNNs and its utilization in time series, as well as the implementation of deep learning for seismic analysis.

2.1 Deep learning on complex data

Traditional machine learning often requires considerable effort from the user to construct meaningful features, which usually is rather time-consuming and error-prone [8]. Deep Learning provides a way for automatic feature extraction with help of multiple layers that can utilise nonlinear processing. In particular, this also relates to complex representations such as multivariate time series and graphs. Therefore, deep

learning offers strong processing and learning on complex data.

Initially, the multilayer perceptron (MLP) was developed in which all network layers are fully linked [18]. While being powerful, due to its high computation time the depth of the network is limited. Therefore, researchers have found ways to create more advanced architectures for specific tasks. One of the most prominent and successful outcomes of this effort is the CNN.

A CNN is a regularized MLP that is specialized in handling data structures with multiple dimensions (e. g., pictures with color channels). It uses a feed-forward structure with convolutions instead of more general matrix multiplications. CNNs have been widely adopted in natural language processing and computer vision. A CNN has an advantage over MLPs due to its use of weight sharing, sampling and local receptive fields [19].

For creating output, the convolutional layers convolve the input using filters and activation functions. A convolution operation is defined as

$$y_i^{l+1}(j) = k_i^l \cdot M^l(j) + b_i^l, \tag{1}$$

where $y_i^{l+1}(j)$ denotes the input of the j-th neuron in the feature map i of layer $l+1$, k_i^l the weights of the i-th filter kernel in layer l, $M^l(j)$ the j-th local region in layer l and b_i^l the respective bias. An activation function is applied after each convolutional layer to retrieve the nonlinear features.

2.2 Graphs

Before discussing the extension of deep learning models to graphs, we first introduce some background and basic notation. We define a graph G as $G = (V, E)$ where V is the set of nodes and E the set of edges (see Fig. 1b for an example). An edge $e_{ij} = (v_i, v_j)$ connects two nodes $v_i, v_j \in V$. A common way to represent a graph is with an adjacency matrix $A \in \mathbb{R}^{N \times N}$ where $N = |V|$, which is a square matrix such $A_{ij} = 1$ if there is an edge from node v_i to node v_j, and 0 otherwise. The number of neighbors of a node v is known as the degree of v and is denoted by $D_{ii} = \sum_j A_{ij}$, where D is then the diagonal degree matrix. Edges can be undirected and directed. Undirected edges contain no notion of source and destination, e. g., the absolute distance between two nodes is always equal no matter from which node the measurement starts. Directed edges do contain direction information, e. g., whether somebody follows someone else on a social network or not.

In addition, nodes and edges (as well as entire graphs) can have *features* as well, such that a feature vector $a = (a_1, a_2, \ldots, a_n)$ of individual features $a_i \in \Omega_A$ out of a feature domain Ω_A is assigned to the nodes (and/or edges). GNN problems therefore mostly consist of node-level, edge-level and graph-level tasks, utilizing the aforementioned feature types.

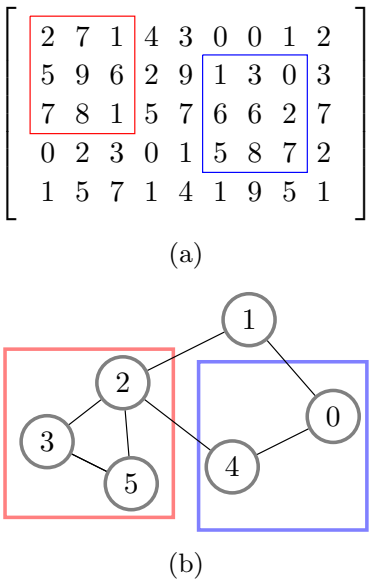


Fig. 1 Examples: **a** matrix-based convolution on an image/time series or **b** a graph, for which a convolution is a lot harder to define than in (**a**)

However, standard convolutional layers (e. g., CNNs) are not applicable to graph-structured data due to its non-euclidean nature. In particular, one cannot convolve an $n \times n$ grid over a graph the same way as with an image. Figure 1 shows an example: both the red and blue boxes convolve over the same grid with 3×3 numbers (Fig. 1a). The red box convolves three nodes, while the blue box convolves over two nodes as shown in Fig. 1b. Thus, extensive effort was put into finding ways to define convolutions over graphs.

2.3 Graph neural networks

GNNs are deep learning-based methods that are adapted for the graph domain. In general, the history of creating deep learning models for graphs is surprisingly long. For example, recursive neural networks were already adapted to work on directed acyclic graphs in the 1990s [20]. However, one recent paper revamped the interest in using deep learning on graphs [21]. They propose two ways that use hierarchical clustering and the spectrum of the graph Laplacian to perform convolutions on low-dimensional graphs. The approach from [21] falls into one of the two historical main methods to perform convolutions with graphs: (1) spectral methods and (2) spatial methods.

Spectral methods use the eigenvectors and eigenvalues of a matrix with eigendecomposition, and perform convolutions using the graph Fourier transformation and the inverse graph Fourier transform, respectively. These transformations of the signal x are defined as $F(x) = U^T x$ and $F^{-1}(x) = Ux$,

where U represents the matrix of eigenvectors of the normalized graph Laplacian $L = I - D^{-1/2}AD^{-1/2}$, D is the degree matrix of the adjacency matrix A and I refers to the identity matrix of length $|V|$ [22].

Spatial methods use *message passing* techniques, which consider the local neighborhood of nodes and perform calculations on their top-k neighbors. With a node aggregation/update function f, an updated node representation Z could then be defined as $Z = f(G)X$ where G refers to the adjacency or Laplacian matrix, and X to the node features of the nodes contained in G [23]. However, a serious issue with spatial methods is in determining the convolution procedure with differently sized node neighborhoods [22].

To conclude, there are two typical operations when designing GNNs: Spatial methods focus more on the connectivity of the graph, while Spectral methods focus on its eigenvalues and eigenvectors [23]. Both approaches were then simplified by Kipf and Welling [24] into the so-called graph convolutional networks (GCNs), which are also used in this paper. They define their propagation rule (convolution in a graph) as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

where $H^{(l)} \in \mathbb{R}^{N\times D}$ is the matrix of activations of the lth layer, σ denotes the selected activation function, $\tilde{D} = \sum_j \tilde{A}_{ij}$ refers to the degree matrix; matrix $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph G with the added self-connections I_N to include a node's own node features, $H^{(0)} = X$ where X are the node features and $W^{(l)}$ is the trainable weight matrix for a specific layer.

A different method was proposed by [17] (graph attention networks, now referred to as GAT), where the structural information of A is dropped and is more implicitly defined by using self-attention over the node features. The authors motivate this by referencing previous work (e. g., transformers [25]) that showed that self-attention is sufficient. Still, both techniques (GCNs and GATs) can produce node-specific outputs of $N \times F$ features, where F is the number of desired output features for each node N. Based on this, we will discuss extensions for time series analysis below.

2.4 Graph neural networks for time series analysis

Considering the connection between GNNs and classical time series analysis, most effort is visible in time series forecasting [10,26]. These approaches adapt existing neural network architectures to use operators from the graph domain. Examples are gated recurrent GNNs that utilise the spectral convolutions from [27]. Also, Diffusion-Convolutional Networks are introduced that take into account the in and out-degree of nodes to capture the spatial dependencies of nodes better, which is beneficial in e. g., traffic prediction [28]. Later on, spatiotemporal graph convolutional neural networks are introduced that interchange the convolution procedure between the temporal and spatial dimensions [29]. In addition, GNNs have been used to perform anomaly detection in time series data. [11] propose an attention-based GNN that used the results of a forecast to classify deviating predictions as anomalies. In addition, [12] propose GRIL (graph recurrent imputation layer), a spatial-temporal GNN that reconstructs missing data by learning spatial-temporal representations.

To conclude, a lot of progress has been made in combining GNNs with classical time series-related tasks. However, time series regression (and classification) tasks have not received the same amount of attention yet. Especially when the target value does not rely on more recent values from the input, but rather on the whole length of the time series, other model architectures are needed. As discussed below, seismic data are a typical domain where these data characteristics naturally arise.

2.5 Deep learning for seismic analysis

Over the past decades, huge volumes of continuous seismic data have been collected [30,31]. With the availability of large datasets and advances in machine learning, the seismological community has also seen a rise in the use of machine and deep learning. Exemplary use cases are magnitude estimation [32] and earthquake detection [33]. Specifically for waveform analysis, the CNN has been applied several times. For example, [34] developed a CNN for single-station localization, magnitude and depth estimation. In addition, CNNs were developed for P- and S-wave arrival times picking [33,35]. Others used multistation waveforms which were analyzed for the estimation of the earthquakes' location [5,36] or early warning [37].

The work of [5,13,14] comes most close to the goals of our work. Each of these papers tried to use time series-related data in combination with graphs to improve predictions. [5] used classical CNNs that attached the (latitude, longitude) locations of the sensors to the waveforms to improve predictions, which differs from our goal to use GNN layers. In other words, metadata was used to enhance their CNN model with spatial-information, except no graph layers were applied. [13] propose a technique that combines CNNs with graph partitioning to group time series together based on spatial information. This procedure increases the quality of the within-group features, and improves predictions, but no GNNs were utilised. Another recently proposed method for handling seismic data with GNNs is from [38]. Here, the location and magnitude of earthquakes is predicted. However, their input are pre-calculated characteristics of the earthquakes for each station, which differs from our goal to use raw

waveform data. Lastly, [14] present a method that uses CNNs and GNNs for seismic event classification. However, no spatial information is provided to the model (i.e., adjacency matrices only containing 1's are created), no meta information of the nodes is added, only three stations are examined simultaneously, and their method focuses on time series classification instead of regression.

In this paper, we specifically propose a technique that will use the full power of GNNs to perform time series analysis. To the best of our knowledge, no GNN-based method was used to perform such a time series regression task before.

3 Method

In this section, we first introduce the datasets used in the experiments. With this as context information, we then define our problem formally. After that, we describe how to generate networks given the datasets. Next, we present the framework of our proposed model for multivariate time series regression (TISER-GCN). We first provide an intuition on an abstracted level, followed by a detailed discussion of the full architecture shown in Fig. 5, and its implementation. Finally, we discuss model training and the applied baseline models which are used for our evaluation.

3.1 Dataset

We perform regression on two datasets recorded by the Italian national seismic network [39,40], described fully in [15,16]. GNNs are an ideal candidate for the analysis of seismic data, since seismic measurements contain (1) an enormous amount of data and (2) sensors that are geographically grounded. Each sensor (i.e., seismometer or accelerometers) in the dataset continuously records the amplitudes of the seismic waves resulting from earthquake occurrences along three components of ground motion (i.e., 3 dimensions): up-down, north-south, and east-west. The input maximum (i.e., the greatest amplitude detected across all stations during the time window) is used to normalize the data, as performed by [15]. The data recorded by the sensors located at the stations are crucial for seismologists to understand the nature of the recorded earthquakes (e.g., magnitude, location, focal mechanism, etc.).

Because information (via telecommunication) can be transmitted faster than the seismic waves travel, seismologists have developed algorithms to predict the maximum intensity measurements (IMs) of ground shaking at a set of far-away stations, caused by an earthquake, using only the very first stations that recorded the earthquake already. In the seismological literature, this objective is known as "earthquake early warning". The IMs used here include *peak ground acceleration* (PGA), *peak ground velocity* (PGV) and *spectral acceleration* (SA) at 0.3, 1 and 3 s periods and represent the labeled data of our model.

Therefore, the task with these datasets is as follows: by using the earthquake recordings from the stations nearby the epicenter, recorded within 10 s from the origin time of the earthquake, we make predictions of the IMs at all stations within the network. A large majority of the stations have not yet recorded the maximum earthquake-related ground motion or ground motion at all. Therefore, we hypothesize that GNNs are highly suited for this time series regression task to predict IMs.

An example of an earthquake (red solid star) drawn from our dataset is shown in the left of Fig. 2. After the initial earthquake waves start to spread, only one station (called FEMA) has started recording part of the earthquake, while the other stations farther to the northwest have not recorded the first waves.

The CI dataset consists of 915 earthquakes recorded on a set of 39 stations (CI network) in central Italy. The earthquake epicenters and station locations are within the area that consists of latitude [42°, 42.75°] and longitude [12.3°, 14°], with earthquakes happening from 01/01/2016 until 29/11/2016. It contains many spatially concentrated earthquakes and a dense network of stations. Earthquakes have a depth between 1.6 km $\leq$ z $\leq$ 28.9 km and magnitudes in the range 2.9 $<$ M $\leq$ 6.5.

The CW dataset consists of 266 earthquakes recorded on a set of 39 other stations (CW network) in central-western Italy. The earthquake epicenters and station locations are within the area bounded by latitudes [41.13°, 46.13°] and longitudes [8.5°, 13.1°], with earthquakes spanning the time period between 01/01/2013 and 20/11/2017. All the earthquakes are in the depth between 3.3 and 64.7 km, with magnitudes in the range 2.9 $<$ M $\leq$ 5.1. Therefore, the CW dataset clearly covers a larger area than the CI dataset, and as Fig. 3 illustrates, the earthquakes of the CW dataset are scattered across a large part of central and northern Italy, whereas the CI dataset has earthquakes concentrated in one small area.

3.2 Problem definition

The goal in this work is to regress various values from multivariate time series sensor data. We test our models on seismic data, where the maximum intensity measurements of shaking at each station should be predicted. We calculate values that are external to the input and do not depend necessarily on recent values, but rather on the whole length of the time series (see Fig. 2). Let L be a symmetrically normalized laplacian matrix $L \in \mathbb{R}^{N \times N}$ where N refers to the number of nodes in the graph, and Z be a node feature matrix $Z \in \mathbb{R}^{2 \times N}$ that holds the latitude and longitude location of each node. Given the input time series $X \in \mathbb{R}^{E \times N \times T \times C}$ where E is the number of earthquakes, N the number of stations, T the length

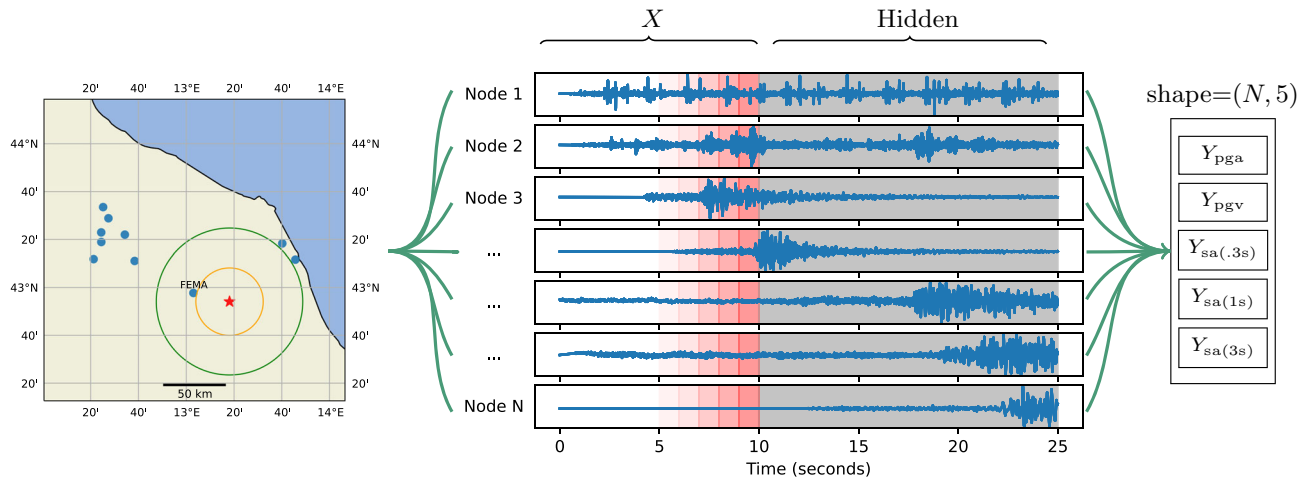


Fig. 2 Overview of the task tackled in this paper. An example earthquake (P (green) and S(orange) wavefronts after 10 s from the origin time) is shown on the left as red star. By taking the initial input length (10 s) of X, we predict the Y values that characterize the earthquake at each node (representing a seismic station). Y has to be inferred by exploiting waveform patterns in X, since Y mostly reveals itself later on in the hidden part of the data (and do not necessarily occur at peaks). In addition, there could be e. g., noise or sensor malfunctioning hindering information. We reduce the 10 s window length progressively by 1 s (each red block) at the time, to further complicate the task in Sect. 4.1.1

of the time series and C the amount of channels, our goal is to predict $Y \in \mathbb{R}^{5\times N}$, which refers to the 5 target parameters of the time series called PGV, PGA, SA(1 s), SA(0.3 s) and SA(3 s) for each node in the graph. The task is iteratively complicated (see Fig. 2) by reducing the input length T given to X in Sect. 4.1.1.

Our final regression problem can then be formulated as follows:

$$f : L \times X \times Z \rightarrow Y \tag{3}$$

where f denotes the learning function, L the graph, X the time series input, Z the node features and Y the regression targets.

3.3 Network creation

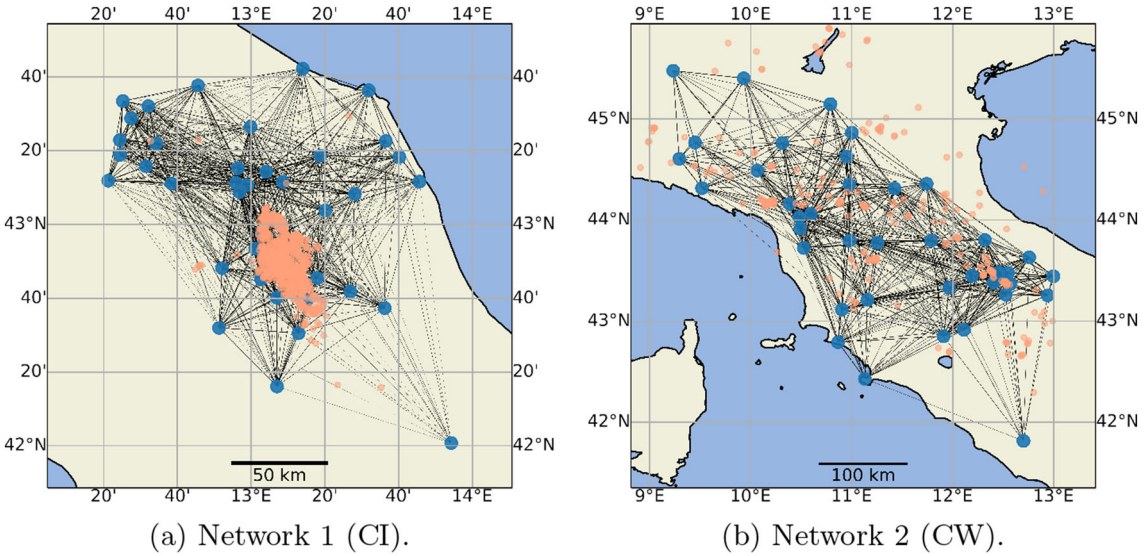
Both undirected sensor networks were created by making use of the geographical locations of the seismic sensors. The adjacency matrix $A_{i,j}$ was calculated by taking all the pairwise geodesic (the shortest path between two points on a sphere) distances in km between each station (latitude, longitude), and taking the $1 -$ (min,max) scaled distance as the edge weight, since edges with a low distance should have a higher weight. Afterward, the resulting adjacency matrix can be filtered on the threshold k to adjust the sparsity in the graph, e. g., the higher the parameter k is set, the fewer edges will retain in the graph (k has a range between 0 and 1). Experimentation showed that in the CI network a threshold of 0.3 was most optimal, and 0.6 in the CW network (see Sect. 4.2.1).

The adjacency matrix, however, still has to undergo some more changes to make it more suitable for GNNs (especially GCNs). Therefore, we transform the adjacency matrix A into the symmetrically normalized Laplacian matrix $L = I - D^{-1/2}AD^{-1/2}$ where D refers to the Degree matrix containing the neighbors of each node and I refers to the identity matrix of length n nodes in a graph. A typical Laplacian would only consist of $L = D - A$, however, if nodes have a wide range of varying connectivity, vanishing gradient problems can occur [24]. Therefore, the degree matrix is symmetrically normalized. Lastly, the addition of the identity matrix helps with the GNN to also involve each node's own node features [24].

Figure 3 visualizes the resulting graphs; the nodes resulting from the seismic stations of the CI and CW datasets are shown in panels a) and b), respectively. As mentioned before, looking at the geographical maps and the coordinates (latitude and longitude) on the axis of the plots, it is clear that the CW network covers a larger land area. In addition, the thickness of the edges in the figure is determined by the distance between two stations. The less distance between two stations, the higher the edge weight. A higher weight will help the GNN with determining which stations will most likely inhibit similar behavior in their sensor readings, improving the IMs' prediction.

3.4 Abstracted framework

Figure 4 presents an abstracted overview of the building blocks of our proposed architecture, which can therefore also



(a) Network 1 (CI). (b) Network 2 (CW).

Fig. 3 Overview of the source-receiver geometries of the CI (**a**) and CW (**b**) seismic datasets. The blue solid dots correspond to the seismic stations (nodes) and the orange dots refer to the earthquake epicenters. Notice the larger geographical area and the sparseness of the epicenters of the CW dataset when compared to CI (visible in the map scale at the bottom of both figures). The thickness of the lines connecting the stations (i.e., the edges) are inversely proportional to the distance of the connecting nodes as from the values of the adjacency matrix

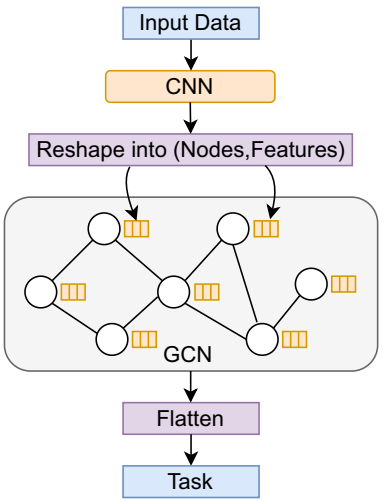


Fig. 4 Abstract overview of our GNN implementation for multivariate time series processing

be instantiated for other tasks, such as time series classification. In summary, our proposed architecture of a GNN for time series regression (TISER-GCN) contains the following main contributions compared to previous work, as we will detail below:

1. To obtain node features, we apply a 1D convolutional layer for feature extraction on the individual nodes using a wide kernel [8,41] on the input data as in [15].
2. To obtain the graph, the set of stations in the seismic area are considered as nodes, with the distance between them as edges.
3. A GNN (utilizing GCN layers from [24]) of n layers is implemented for processing these feature vectors calculated by the convolutional layers as node features. While in other GNN papers the node features each measure a unique aspect about a node (e. g., the age and friend count in a social network), we demonstrate that GCNs can also learn from features that are sequential in time.
4. In our case, as described below, we focus on a *regression task* on seismic data, focusing on predicting ground shaking at a set of seismic stations.

3.5 Model implementation

This section introduces the version of our abstracted implementation applied to a regression task on seismic data. For providing a complete picture of the model, the source code is available.[1]

The first block of our proposed model uses as input 10 s of 3-channel seismic waveform data sampled at 100 Hz, i. e., a time series from each station of each earthquake. See Fig. 5 and Sect. 3.1 for a full overview of the model and the dataset. After that, convolutional, graph-convolutional and post-processing layers are applied.

[1] https://github.com/StefanBloemheuvel/GCNTimeseriesRegression.

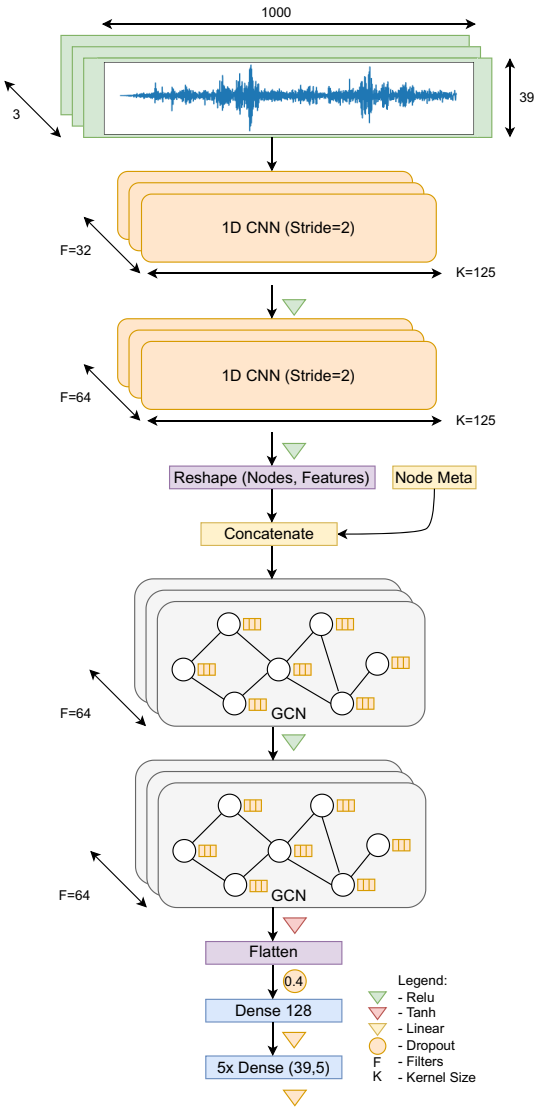


Fig. 5 Overview of the proposed architecture. The features from two convolutional layers are used as node features in the GCNs. After the two GCN layers (which are used for inter-station-related feature extraction), the data are flattened to retain as much information as possible. Afterward, the output is fed to fully connected layers

3.5.1 CNN for feature extraction

In the second block of our model, two 1D convolutional layers act as feature extractors by using wide kernel sizes, small strides, increasing filters, kernel regularization and a ReLU activation function, which has proven to be useful for 1D time series data [15,41]. The purpose of these convolutional layers is to learn the temporal patterns of each station. Afterward, the output of the second convolutional layer, which has shape (N, T, F) where N refers to the number of nodes, T the remaining length of the time series and F the number of filters, is reshaped to make the dimensions fitted for the graph convolutional layers. These layers typically need an input of (N, F) where F now refers to a one-dimensional vector $[x_1, x_2 \ldots x_n]$ for each node in the graph.

To this reshaped feature vector, features (latitude, longitude) of each node are added as node meta data. Therefore, the feature vector of each node now consists of time series features from the convolutional layers and classical node features. This addition of this node metadata has showed to improve performance in [16].

3.5.2 GCN processing

Next follows the graph convolutional layers used from [24]. While in [15] the third convolutional layer gathers the cross-station information, here the graph convolutional layers take this role, since they use the features from the convolutional layer as node features for each node. More concretely, each node N receives one of the feature vectors of dimension (N, F) as node features where F is the length of the feature vector (see Fig. 4). The two graph convolutional layers use these features of the nodes by reducing them to $(N,64)$ by both containing 64 filters. Considering the hyperparameters, experimentation revealed that starting with a ReLU activation function followed by a TanH works best. In addition, bias was set to false (as suggested by [24]) and the same kernel regularizer was used as in the convolutional layers.

3.5.3 Postprocessing

A common practice in the graph literature is using global graph pooling operators such as max or average-pooling [27,42,43]. These pooling techniques take the embeddings of all the nodes in a graph and globally pool these together by an aggregation function (max, sum, mean ...) However, this procedure reduces the feature vector from (N, F) into a single vector F regarding the number of nodes and filters used in the previous layer. This means that the graph is reduced to essentially one node, which is not desirable in our task, because this is defined as a node-level regression task for all the nodes in the graph, in contrast to a graph-level task [42]. Therefore, the output of the final graph convolutional layer is directly flattened and then fed to the fully connected layer.

The output of this layer is then given to five fully connected *regression* layers. These layers represent the regression target variables called PGV, PGA, SA(0.3 s), SA(1 s) and SA(3 s) for each of the nodes.

3.6 Software and computer

Python was used in combination with Tensorflow[2] and Keras[3] to develop the proposed models. The GCN layer is

[2] https://www.tensorflow.org/.

[3] https://keras.io/.

derived from Spektral.[4] Calculations are done with support of Numpy[5] and table formatting with Pandas.[6] Furthermore, to reduce the overall training time, the models are trained on a dedicated server with two Intel Xeon CPUs (3.2 GHz), 256 GB RAM and a Nvidia Quadro RTX3956000 (24 GB) GPU. After training, the models are fairly small (around 6 MB) and are deployable, e. g., on standard PC hardware as well as edge computing platforms.

3.7 Model training

Regarding model training, 80% of each dataset was used for training and 20% for testing. The train set was then randomly split by k-fold cross-validation with $k = 5$. The average MAE, MSE and RMSE scores on the test set were taken as the results. This entire procedure was repeated 5 times with different seeds, which generates 5 distinct train-test splits to generalize the results (i.e., the large amplitudes from the less frequent large earthquakes in a test set can influence the scores if more of them are assigned to a test set).

The model used a batch size of 20 and 100 training epochs with early stopping—patience of 10. The same optimizer as in our baseline [15] was used, namely RMSprop with mostly standard settings [44]. Lastly, MSE (mean squared error) was used as the loss function when training the models:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2, \tag{4}$$

where y_i are the actual values and $\hat{y}_i$ the predictions, since it penalizes larger errors more than e. g., Mean Absolute Error

3.8 Baseline models

We compare our work to the model presented by [15]. This baseline model was given the exact same input data except for the graph, i. e., both models received the time series per station for every earthquake combined with the latitude and longitude node features of every station, as proven to be effective in [16]. In addition, we also examine the performance of GAT [17] layers and an adjusted version of [14] for our task. The GAT layers are set with 8 channels and 8 attention heads in order to match the number of parameters of our model, all other hyperparameters were unchanged. To adapt the model of [14] to our task, the last layers used for classification were altered for regression, a weighted initial adjacency matrix was supplied, and node features were added.

[4] https://graphneural.network/.

[5] https://numpy.org/.

[6] https://pandas.pydata.org/.

Table 1 Overview of possible parameter settings used for the grid search optimization of the ML models

Model	Parameter	Option range
K-NN	K	1–20 (1 per step)
	Weight options	Uniform or distance
SVM	C	10–40 (5 per step)
	Gamma	[0.0001,0.001]
	Kernel	Linear or radial basis function
XGBoost	*N* Estimators	100–1000 (100 per step)
	Max depth	5–15 (5 per step)
	Gamma	[0.0,0.1,0.2,0.3,0.4]
RF	*N* estimators	100–1000 (100 per step)
	Max features	Square root or Log2

In addition, our proposed model is also compared with traditional machine learning (ML) algorithms: k-nearest neighbors (K-NN), extreme gradient boosting (XGBoost), random forest (RF) and support vector machines (SVM). Since these models are not designed to process multidimensional data, features were calculated from both the time and frequency domain. These features are derived from several studies [45,46]. From the time domain, the mean, standard deviation, variance, median, minimum, maximum and range (maximum–minimum) are used. From the frequency domain, the signal energy $E = \sum(\mathrm{fft}\, x_i)^2$ and signal power $P = \sum \frac{(\mathrm{fft}\, x_i)^2}{\sum_i^I}$ were used. For each of the ML models, grid search optimization in combination with fivefold cross-validation was used to assess which models performed best. Table 1 describes all the options of the grid search optimization.

Lastly, we also compare the best performing deep learning models for each network without node features added, to examine their effect on the performance. These results are visible in the Ablation study in Sect. 4.2. Here, the impact of the spatial information can be observed.

4 Results

In this section, the results from the multivariate regression task are shown. In addition, our model is tested on different input window lengths to further complicate the task (Fig. 6).

4.1 IM prediction

The results of the IM prediction are visible in Table 2 and Figs. 7 and 8. All the algorithms perform better on the CI network than on the CW network. Such behavior is expected since the CW dataset contains fewer earthquakes (266 against 915) than the CI dataset and, in contrast with the CI, their spa-

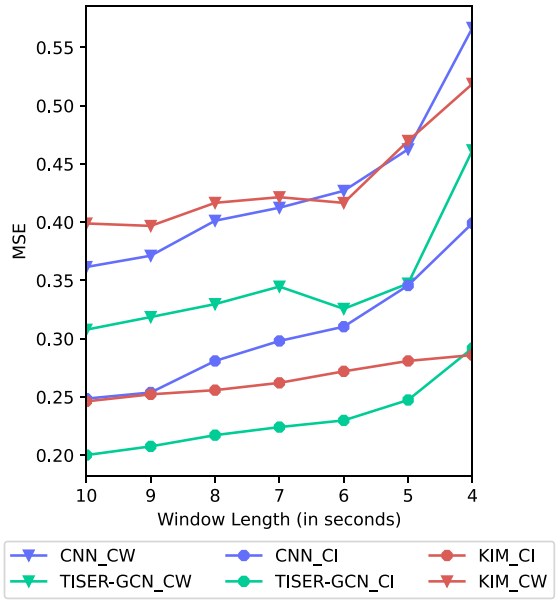


Fig. 6 MSE at different input window lengths for each model in both networks. Read the figure from left (initial 10 s window from Sect. 4.1) to right. Our model is capable of achieving approximately the same MSE scores when given half the input, highlighting the power of GCNs with processing spatial information

tial distribution is sparse. In addition, the CW network covers a larger area, with greater distances between the stations, a larger depth range of the hypocenters and a larger variability in the geological settings. As an extra test, 266 samples were taken from the 915 earthquakes in the CI dataset to mimic the conditions of the CW dataset while preserving the densely located earthquakes characteristic of the CI dataset. In general, this resulted in an increase in MSE of 30% for our model (TISER-GCN) and 42% for the CNN model from [15], showing the importance of having enough samples for learning.

When examining the individual performance of the models, TISER-GCN outperforms the best performing baselines (the model from [14] on the CI network and the CNN model from [15] on the CW network, respectively) by a large margin on each of the five metrics of ground motion. Especially in the PGA and SA(1 s) metrics this performance gain is visible. Our model improves an average of 7% on MAE, 16.1% on MSE and 8.3% on RMSE compared to the best baseline for the CI network. Considering the CW network, an improvement of 9.1% on MAE, 16.5% on MSE and 8.4% on RMSE is achieved.

Lastly, it is interesting to see the relatively weak performance of the GAT-based model. A possible explanation could be that the explicitly defined spatial information in the graphs (the distances between the stations) is crucial to make sense of the time series data, which the GAT layers infer themselves via self-attention. Therefore, perhaps the time series features are too complex for the GAT layers to learn such representations, especially in the CW network.

Considering the results of the models on each individual earthquake and each IM metric, Fig. 7 shows the observed versus the predicted IM values of the CW network for both [15] and TISER-GCN as residual plots. The blue lines (and residuals) reveal the performance of the CNN model [15] and the green lines (and residuals) TISER-GCN. The lines were calculated with an ordinary least squares to better visualize the difference in prediction bias between the two models. We observe, that better performance of our model comes also in slight reduction of the bias for large IM values (less underestimation), which is also of great value for the seismological applications (for more information, see the original CNN model [15]).

Lastly, the characteristics of almost all the deep learning models are highly similar. The models do not deviate concerning Ms/step per epoch. However, there is a small decrease in the number of parameters, since our TISER-GCN model has 6.7% fewer parameters than the CNN model from [15]. The GAT model has an equal amount of parameters as our model, and the adapted GCN model from [14] originally already has fewer parameters.

4.1.1 Variation in different window lengths

Section 4.1 shows the results based on a 10 s input window length. However, it is interesting to investigate the results of both the CNN and the best performing GNN-based models when this window length is reduced, since smaller window lengths could translate in earlier responses. However, a smaller input length creates a more difficult setting, since fewer stations have received enough information to predict the IMs at further away stations. Therefore, all the window lengths between 10 and 4 s (it is not reasonable to reduce the input window further with this specific application) and their corresponding MSE scores are displayed in Fig. 6.

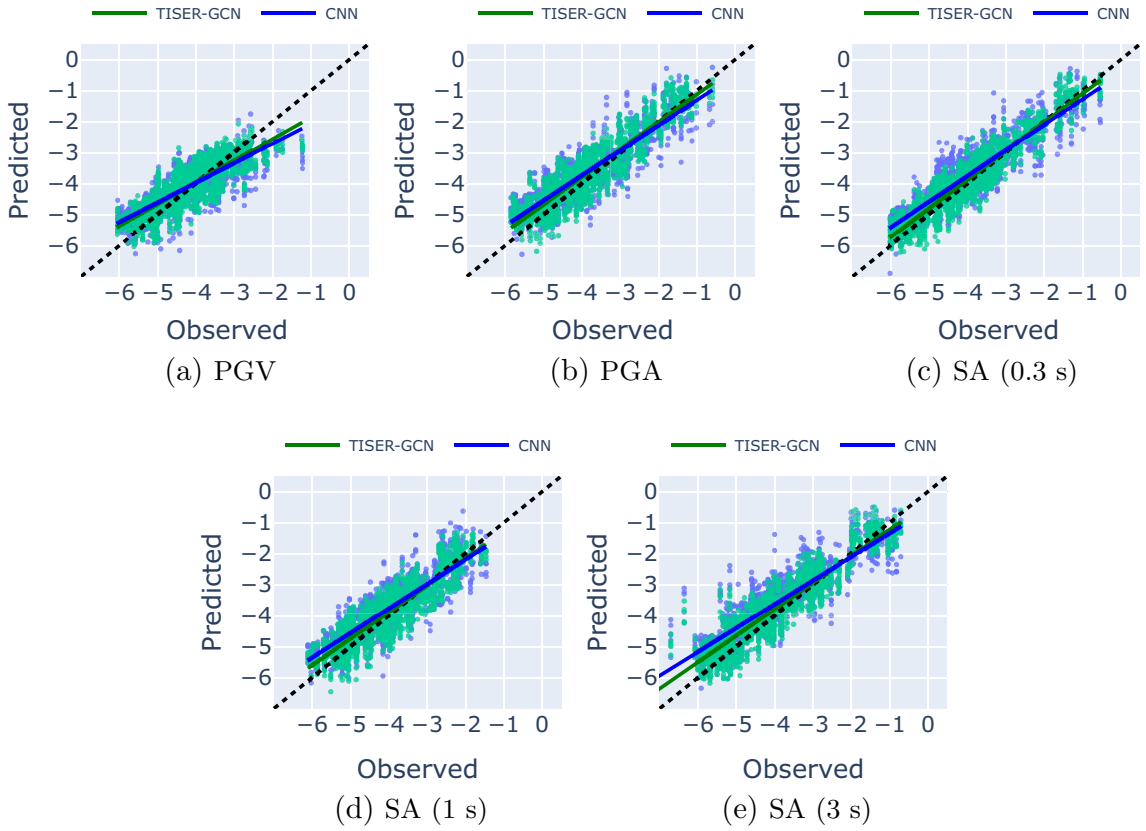
We find that we can half the entire input window, while still achieve similar performance as the best performing baselines. This reduction also results in a reduction of the model parameter size of 1.26 million to around 700 thousand (−44%).

Overall, these results highlight the power of the GCN layers and our implementation. Since GCN layers are designed to perform node feature sharing in its convolution procedure, it is still possible to transfer plenty of information between the nodes in a situation where half the input is provided. In the context of analyzing streaming time series data, these benefits are crucial [47] since requiring less input translates to earlier responses.

Table 2 Individual results of each IM metric for the support vector machine (SVM), K-nearest neighbors (KNN), XGBoost, random forest (RF), the CNN model from [15], GAT layers, adjusted implementation of [14] and our proposed model (TISER-GCN)

	PGA			PGV			PSA03			PSA1			PSA3		
	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE	MAE	MSE	RMSE
CI network															
SVM	0.43	0.36	0.60	0.47	0.43	0.65	0.47	0.41	0.64	0.44	0.37	0.61	0.45	0.40	0.63
KNN	0.41	0.32	0.56	0.44	0.37	0.61	0.45	0.37	0.61	0.43	0.35	0.59	0.44	0.38	0.62
XGBoost	0.38	0.28	0.53	0.41	0.32	0.57	0.42	0.33	0.57	0.41	0.31	0.56	0.41	0.33	0.58
RF	0.38	0.28	0.53	0.41	0.32	0.57	0.42	0.33	0.57	0.41	0.31	0.56	0.41	0.33	0.57
GAT	0.39	0.30	0.54	0.36	0.26	0.49	0.36	0.26	0.49	0.38	0.28	0.52	0.37	0.28	0.52
Jozinovic et al. [15]	0.34	0.22	0.46	0.35	0.26	0.50	0.36	0.24	0.48	0.35	0.26	0.49	0.36	0.25	0.49
Kim et al. [14]	0.35	0.26	0.49	0.33	0.23	0.47	0.33	0.23	0.47	0.33	0.24	0.48	0.33	0.24	0.48
TISER-GCN	**0.31**	**0.20**	**0.44**	**0.32**	**0.21**	**0.45**	**0.31**	**0.19**	**0.43**	**0.31**	**0.20**	**0.43**	**0.32**	**0.21**	**0.45**
CW network															
GAT	0.54	0.49	0.68	0.56	0.52	0.70	0.55	0.52	0.70	0.53	0.49	0.68	0.58	0.56	0.72
SVM	0.51	0.43	0.66	0.56	0.51	0.71	0.60	0.58	0.77	0.56	0.51	0.72	0.47	0.40	0.63
KNN	0.52	0.45	0.67	0.57	0.51	0.71	0.61	0.60	0.78	0.57	0.53	0.73	0.48	0.41	0.64
XGBoost	0.50	0.42	0.65	0.54	0.48	0.69	0.59	0.57	0.75	0.55	0.51	0.72	0.46	0.39	0.62
RF	0.49	0.40	0.63	0.54	0.47	0.68	0.58	0.56	0.75	0.55	0.50	0.71	0.46	0.39	0.62
Kim et al. [14]	0.45	0.35	0.59	0.48	0.40	0.62	0.46	0.38	0.60	0.45	0.35	0.58	0.46	0.37	0.60
Jozinovic et al. [15]	0.44	0.35	0.58	0.46	0.37	0.59	0.44	0.35	0.58	0.48	0.40	0.62	0.45	0.36	0.58
TISER-GCN	**0.41**	**0.30**	**0.54**	**0.41**	**0.30**	**0.54**	**0.40**	**0.29**	**0.52**	**0.42**	**0.31**	**0.54**	**0.43**	**0.33**	**0.56**

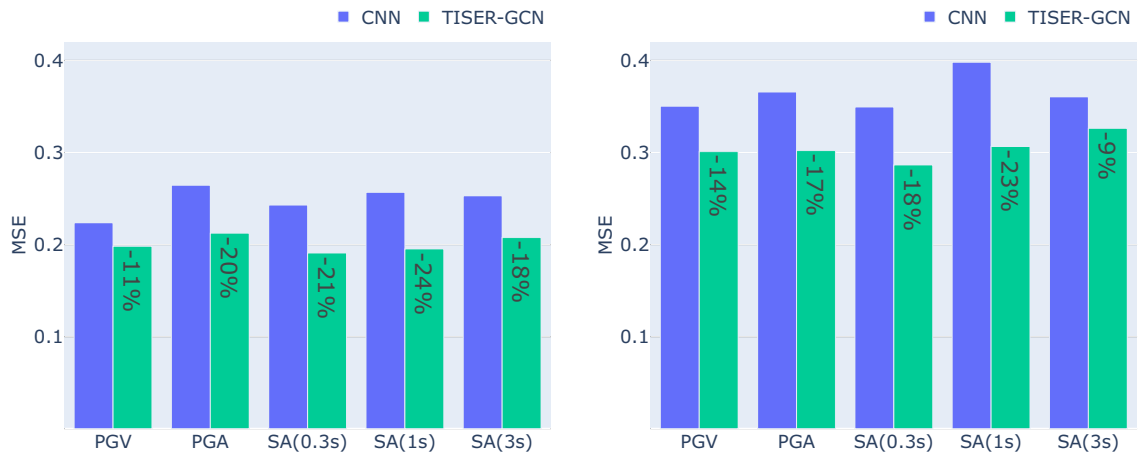
The bold rows refer to the best performing model TISER-GCN (ours)



(a) PGV (b) PGA (c) SA (0.3 s)

(d) SA (1 s) (e) SA (3 s)

Fig. 7 Residual plots of the predicted against the true 5 IMs of the CW dataset (displayed in logarithmic ($\log_{10}$) form). The blue line and points display the results of [15] CNN, the green line and points of TISER-GCN. The dotted black line resembles a perfect prediction score. The units are m/s for PGV and m/s^2 for PGA and SA



(a) Network 1 (CI). (b) Network 2 (CW).

Fig. 8 MSE scores for each metric where the blue bar represents the original CNN model by [15] (our initial baseline) and the green bars our proposed model

4.2 Experimentation: ablation study

4.2.1 Tuning hyperparameter k

The graph creation algorithm mentioned in Sect. 3, used the hyperparameter k to cutoff connections between stations that are too far away. k falls between 0 and 1, where 0 means that no edges are filtered, and 1 means that all edges are filtered. Table 3 presents the results of our model when tuning this parameter on both networks and at what distance the cutoff will be. If the MSE scores of two values of k would be highly similar, one would choose a higher k over a lower k, due to the computational advantages of sparser graphs.

While in the CI network the results for k are more promising with lower values of k, different results are visible in the CW network. An explanation can be found in the very different characteristics (see Sect. 3.1) of the two networks. That is, a too low cutoff has bigger effect applied to a widely spaced network (i. e., CW) than on a more concentrated network (i. e., CI). In practice, having more edges than necessary in a largely spread network (CW network) apparently confuses the GCN layers in this experiment, perhaps since stations in less dense networks show different behavior sooner than in more densely networks (e. g., the CI network). These results highlight the importance of this preprocessing step when designing graphs. However, it is an easily interpretable hyperparameter that once determined can help GCNs achieve great performance.

4.2.2 Effect of node metadata

The impact of the node features can be examined by inspecting the results if no features (latitude, longitude) were supplied. Without node features, which have proven to be crucial in our difficult prediction task [16], the CNN from [15] and TISER-GCN score approximately equal. Both have an MSE of 0.26 on the CI network, and the GCN from [14] achieves a promising 0.24 MSE. On the CW network, [15] scores 0.50, TISER-GCN 0.51, and the GCN from [14] scores a 0.37. These scores show how difficult this task is without including the spatial information from the stations, which shows to be crucial for this task, confirming the insights from [16]. In addition, they show the promising results from [14] model with no metadata added.

However, once the features are added, changes in performance become visible. The model from [15] improves 5% on the CI network and 26% on the CW network, whereas our model improves 24% on the CI network and 41% on the CW network. In contrast, the GCN from [14] does not improve at all, staying at 0.24 and 0.37 MSE, respectively. Therefore, our TISER-GCN appears to be more capable of using this spatial information to improve the learned feature representation, by combining the graph information with the time series data while exceptionally well exploiting the information contained in the spatial metadata in the graph convolutional layers. However, we want to emphasize that the model from [14] was not originally optimized for this task, but for earthquake classification.

5 Conclusions

In this work, the use of multivariate time series regression with graph neural networks was presented. Our method (TISER-GCN) proposed a unique way to leverage features from convolutional layers as node features in a GCN. The proposed model is tested on two seismic datasets with different characteristics, demonstrating the generalizability of the model. Our model outperforms the best performing baselines by 16.1% on average on the CI dataset and 16.5% on the CW dataset in terms of MSE. Therefore, besides the original baseline of [15], the adjusted version of [14] and GAT lay-

Table 3 Ablation results of the minimum distance cutoff hyperparameter k for both networks

Network	k	Cutoff (km)	Edges	Avg. degree centrality	MSE
CI	0.6	92	497	0.67	0.224
	0.5	115	609	0.82	0.209
	0.4	138	687	0.92	0.191
	0.3	159	722	0.97	0.189
	0.2	177	729	0.98	0.188
	0.1	204	733	0.99	0.191
CW	0.6	217	493	0.67	0.307
	0.5	271	601	0.81	0.314
	0.4	325	660	0.89	0.310
	0.3	377	705	0.95	0.322
	0.2	429	731	0.99	0.360
	0.1	485	739	0.99	0.368

ers [17] also showed weaker performance compared to our model. The experiments demonstrate the impressive power of TISER-GCN, i.e., of our proposed GCN model when processing spatial information, since the tested deep learning models were provided with the same input. More crucially, especially when taking into account the use case of early warning systems, our model can match the performance of the baselines by using half of the input window length on both datasets. Such a reduction can help with faster earthquake early warnings (half the input means a faster response).

One important message which we want to emphasize is that the architecture of the original CNN model by [15] could also be improved from an only CNN perspective. However, to make the comparison more fair, and to observe more directly the actual effect of the GCN layers on the prediction results, we decided to alter the model architecture as little as possible to demonstrate the difference in performance between a classical CNN approach compared to our proposed model.

For future research, other methods of creating the initial adjacency matrix will be investigated, because the results of the cutoff parameter experiments in the ablation study reveal the effect the graph creation steps have on the performance in the CW network. Examples include; to keep the top-k edges for each node or using exponential decaying functions as in [48]. Furthermore, other node features could be added to the node feature vector to improve predictions. For example, the angle between two stations could be added as an edge feature, or the absolute distance between two stations can be used. Here, also explanation techniques [49,50] could yield interesting insights, in order to lead feature construction and modeling.

In addition, we plan to test our model on other (types of) datasets. For example, both networks now consisted of 39 nodes. It would be interesting how our architecture would scale to datasets featuring 200 or more nodes. Also, to test the transfer learning capabilities of our model, perhaps adding the spatial information to a pre-trained model on one dataset could make it more adaptable for other networks with different data characteristics.

Lastly, our architecture was built to be easily adaptable for other tasks (involving regression or classification). Inspiration can be taken from the examples of [7], however, there could be as well many other types of datasets where scalar or vector value quantities, either associated to a single node or to the entire graph are to be predicted. Therefore, we encourage readers to take Fig. 4 as a departure point for other analysis tasks.

Author Contributions SB and MA conceived of the idea and study, as well as the interpretation of the data, which was performed in a synergistic way together with DJ and AM. SB, MA and JH drafted the manuscript. All authors edited the manuscript. SB implemented the methods and algorithms supported by JH, and ran the experiments, in close collaboration with DJ and AM. DJ and AM provided the baseline model and the data. All authors read, reviewed and approved the final manuscript.

Funding This work has been funded by the Interreg North-West Europe program (Interreg NWE), project Di-Plast - Digital Circular Economy for the Plastics Industry (NWE729).
This work was also partially supported by the project INGV Pianeta Dinamico 2021 Tema 8 SOME (CUP D53J1900017001) funded by Italian Ministry of University and Research "Fondo finalizzato al rilancio degli investimenti delle amministrazioni centrali dello Stato e allo sviluppo del Paese, legge 145/2018.

Declarations

Conflict of Interest/Competing interests Not Applicable

Ethic Approval Not Applicable

Consent to participate Not Applicable

Consent for publication Not Applicable

Availability of data and material Data are available at [51] (CI dataset) and [52] (CW dataset).

Code availability There is a Githubpage available with the corresponding code.

References

1. Tilak, S., Abu-Ghazaleh, N.B., Heinzelman, W.: A taxonomy of wireless micro-sensor network models. ACM SIGMOBILE Mob. Comput. Commun. Rev. **6**(2), 28–36 (2002)
2. Tubaishat, M., Madria, S.: Sensor networks: an overview. IEEE Potentials **22**(2), 20–23 (2003)
3. Aslam, J., Lim, S., Pan, X., Rus, D.: City-scale traffic estimation from a roving sensor network. In: Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, pp. 141–154 (2012)
4. Hatchett, B.J., Cao, Q., Dawson, P.B., Ellis, C.J., Hecht, C.W., Kawzenuk, B., Lancaster, J., Osborne, T., Wilson, A.M., Anderson, M., et al.: Observations of an extreme atmospheric river storm with a diverse sensor network. Earth Space Sci. **7**(8), 2020–001129 (2020)
5. van den Ende, M.P., Ampuero, J.-P.: Automated seismic source characterization using deep graph neural networks. Geophys. Res. Lett. **47**(17), 2020–088690 (2020)

6. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
7. Tan, C.W., Bergmeir, C., Petitjean, F., Webb, G.I.: Time series extrinsic regression. Data Min. Knowl. Discov. **35**(3), 1032–1060 (2021)
8. van den Hoogen, J.O.D., Bloemheuvel, S.D., Atzmueller, M.: An improved wide-kernel CNN for classifying multivariate signals in fault diagnosis. In: International Conference on Data Mining Workshops, pp. 275–283 (2020)
9. Ince, T., Kiranyaz, S., Eren, L., Askar, M., Gabbouj, M.: Real-time motor fault detection by 1-d convolutional neural networks. IEEE Trans. Ind. Electron. **63**(11), 7067–7075 (2016)
10. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of KDD, pp. 753–763 (2020)
11. Deng, A., Hooi, B.: Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4027–4035 (2021)
12. Cini, A., Marisca, I., Alippi, C.: Filling the g_ap_s: multivariate time series imputation by graph neural networks. In: International Conference on Learning Representations (2022). https://openreview.net/forum?id=kOu3-S3wJ7
13. Yano, K., Shiina, T., Kurata, S., Kato, A., Komaki, F., Sakai, S., Hirata, N.: Graph-partitioning based convolutional neural network for earthquake detection using a seismic array. J. Geophys. Res. Solid Earth **126**(5), 2020–020269 (2021)
14. Kim, G., Ku, B., Ahn, J.-K., Ko, H.: Graph convolution networks for seismic events classification using raw waveform data from multiple stations. IEEE Geosci. Remote Sens. Lett. **19**, 1–5 (2021)
15. Jozinović, D., Lomax, A., Štajduhar, I., Michelini, A.: Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network. Geophys. J. Int. **222**(2), 1379–1389 (2020)
16. Jozinović, D., Lomax, A., Štajduhar, I., Michelini, A.: Transfer learning: Improving neural network based prediction of earthquake ground shaking for an area with insufficient training data. Geophys. J. Int. **229**, 704–718 (2021)
17. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
18. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
19. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
20. Sperduti, A., Starita, A.: Supervised neural networks for the classification of structures. IEEE IEEE Trans Neural **8**(3), 714–735 (1997)
21. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and deep locally connected networks on graphs. In: 2nd International Conference on Learning Representations, ICLR 2014 (2014)
22. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: a review of methods and applications. AI Open **1**, 57–81 (2020)
23. Chen, Z., Chen, F., Zhang, L., Ji, T., Fu, K., Zhao, L., Chen, F., Wu, L., Aggarwal, C., Lu, C.-T.: Bridging the gap between spatial and spectral domains: a survey on graph neural networks. CoRR (2020)
24. Welling, M., Kipf, T.N.: Semi-supervised classification with graph convolutional networks. In: J. International Conference on Learning Representations (ICLR 2017) (2016)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Adv. Neural Inf. Process. Syst. **30** (2017). arXiv:1706.03762
26. Cao, D., Wang, Y., Duan, J., Zhang, C., Zhu, X., Huang, C., Tong, Y., Xu, B., Bai, J., Tong, J., et al.: Spectral temporal graph neural network for multivariate time-series forecasting. Adv. Neural. Inf. Process. Syst. **33**, 17766–17778 (2020)
27. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Adv. Neural. Inf. Process. Syst. **29**, 3844–3852 (2016)
28. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In: International Conference on Learning Representations (ICLR '18) (2018)
29. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI) (2018)
30. Ingate, S., Husebye, E.S.: The IRIS Consortium: Community Based Facilities and Data Management for Seismology (2008)
31. Strollo, A., Cambaz, D., Clinton, J., Danecek, P., Evangelidis, C.P., Marmureanu, A., et al.: EIDA: the European integrated data archive and service infrastructure within ORFEUS. Seismol. Res. Lett. **92**(3), 1788–1795 (2021)
32. Ochoa, L.H., Niño, L.F., Vargas, C.A.: Fast magnitude determination using a single seismological station record implementing machine learning techniques. Geod. Geodyn. **9**(1), 34–41 (2018)
33. Mousavi, S.M., Ellsworth, W.L., Zhu, W., Chuang, L.Y., Beroza, G.C.: Earthquake transformer-an attentive deep-learning model for simultaneous earthquake detection and phase picking. Nat. Commun. **11**(1), 1–12 (2020)
34. Lomax, A., Michelini, A., Jozinović, D.: An investigation of rapid earthquake characterization using single-station waveforms and a convolutional neural network. Seismol. Res. Lett. **90**(2A), 517–529 (2019)
35. Ross, Z.E., Meier, M.-A., Hauksson, E.: P wave arrival picking and first-motion polarity determination with deep learning. J. Geophys. Res. Solid Earth **123**(6), 5120–5129 (2018)
36. Kriegerowski, M., Petersen, G.M., Vasyura-Bathke, H., Ohrnberger, M.: A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms. Seismol. Res. Lett. **90**(2A), 510–516 (2019)
37. Münchmeyer, J., Bindi, D., Leser, U., Tilmann, F.: The transformer earthquake alerting model: a new versatile approach to earthquake early warning. Geophys. J. Int. **225**(1), 646–656 (2021)
38. McBrearty, I.W., Beroza, G.C.: Earthquake location and magnitude estimation with graph neural networks. arXiv preprint arXiv:2203.05144 (accepted at ICIP 2022) (2022)
39. Michelini, A., Margheriti, L., Cattaneo, M., Cecere, G., D'Anna, G., Delladio, A., et al.: The Italian National Seismic Network and the earthquake and tsunami monitoring and surveillance systems. Adv. Geosci. **43**, 31–38 (2016). https://doi.org/10.5194/adgeo-43-31-2016
40. Danecek, P., Pintore, S., Mazza, S., Mandiello, A., Fares, M., Carluccio, I., Della Bina, E., Franceschi, D., Moretti, M., Lauciani, V., Quintiliani, M., Michelini, A.: The Italian Node of the European Integrated Data Archive. Seismol. Res. Lett. **92**(3), 1726–1737 (2021). https://doi.org/10.1785/0220200409
41. van den Hoogen, J., Bloemheuvel, S., Atzmueller, M.: Classifying multivariate signals in rolling bearing fault detection using adaptive wide-kernel CNNs. Appl. Sci. (2021). https://doi.org/10.3390/app112311429
42. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. Adv. Neural. Inf. Process. Syst. **3**, 1 (2018)
43. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of IEEE ICVPR, pp. 3693–3702 (2017)

44. Hinton, G., Srivastava, N., Swersky, K.: Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on **14**(8), 2 (2012)
45. Mazilu, S., Calatroni, A., Gazit, E., Roggen, D., Hausdorff, J.M., Tröster, G.: Feature learning for detection and prediction of freezing of gait in Parkinson's disease. In: International Workshop on Machine Learning and Data Mining in Pattern Recognition, pp. 144–158. Springer (2013)
46. Masiala, S., Huijbers, W., Atzmueller, M.: Feature-set-engineering for detecting freezing of gait in Parkinson's disease using deep recurrent neural networks. arXiv preprint arXiv:1909.03428 (2019)
47. Domingos, P.M., Hulten, G.: Catching up with the data: research issues in mining data streams. In: DMKD (2001)
48. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. **30**(3), 83–98 (2013)
49. Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., Zhang, X.: Parameterized explainer for graph neural network. Adv. Neural. Inf. Process. Syst. **33**, 19620–19631 (2020)
50. Schwenke, L., Atzmueller, M.: Constructing global coherence representations: identifying interpretability and coherences of transformer attention in time series data. In: Proceedings of the 8th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2021, Porto, Portugal, October 6–9, 2021, pp. 1–12. IEEE (2021). https://doi.org/10.1109/DSAA53316.2021.9564126
51. Jozinović, D., Lomax, A., Štajduhar, I., Michelini, A.: CNNpredIM—dataset for rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network. Zenodo (2020). https://doi.org/10.5281/zenodo.3669969
52. Jozinović, D., Lomax, A., Štajduhar, I., Michelini, A.: Dataset—seismic data from central-western Italy used in the paper on rapid prediction of ground motion using a convolutional neural network. Zenodo (2021). https://doi.org/10.5281/zenodo.5541083