GAP-LSTM: Graph-Based Autocorrelation Preserving Networks for Geo-Distributed Forecasting

Massimiliano Altieri, Roberto Corizzo, *Member, IEEE*, and Michelangelo Ceci

***Abstract*— Forecasting methods are important decision support tools in geo-distributed sensor networks. However, challenges such as the multivariate nature of data, the existence of multiple nodes, and the presence of spatio-temporal autocorrelation increase the complexity of the task. Existing forecasting methods are unable to address these challenges in a combined manner, resulting in a suboptimal model accuracy. In this article, we propose GAP-LSTM, a novel geo-distributed forecasting method that leverages the synergic interaction of graph convolution, attention-based long short-term memory (LSTM), 2-D-convolution, and latent memory states to effectively exploit spatio-temporal autocorrelation in multivariate data generated by multiple nodes, resulting in improved modeling capabilities. Our extensive evaluation, involving real-world datasets on traffic, energy, and pollution domains, showcases the ability of our method to outperform state-of-the-art forecasting methods. An ablation study confirms that all method components provide a positive contribution to the accuracy of the extracted forecasts. The method also provides an interpretable visualization that complements forecasts with additional insights for domain experts.**

***Index Terms*— Forecasting, graph convolution, neural networks, sensor networks, spatio-temporal autocorrelation.**

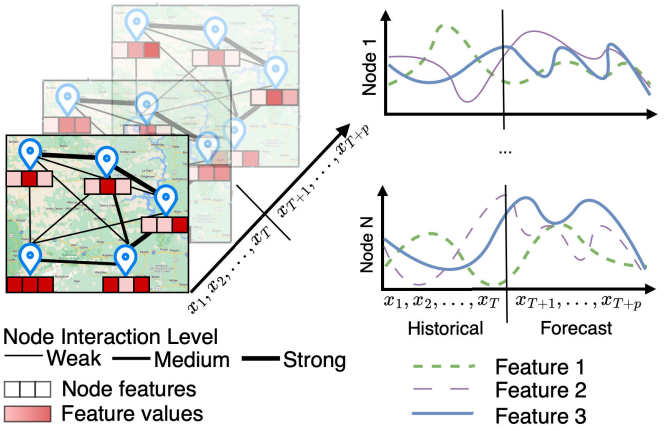

Fig. 1. Overview of the geo-distributed forecasting task addressed by GAP-LSTM. Multiple locations (graph nodes) generate multivariate data (node features) at a fixed time granularity. The model leverages historical data $x_1, x_2, \ldots, x_T$ to extract forecasts for p timesteps. Modeling temporal, spatial, and graph-based interactions is crucial to capture complex and dynamic correlation patterns resulting in accurate multistep-ahead forecasts.

I. Introduction

TIME series forecasting represents a crucial task in many real-world domains, including renewable energy, traffic, and pollution. Indeed, highly accurate forecasting models can be a powerful decision support tool to domain experts, providing the required knowledge to define new policies, foster operational safety, improve resource planning, and increase revenues. In the energy domain, accurate forecasting tools can lead to efficient integration of renewable energy with fossil sources, aiming at a balanced power grid load distribution, as well as effective energy trading strategies. In the traffic domain, forecasting tools can support traffic redirection policies to effectively distribute traffic across roads. In the pollution domain, they can help improving air quality in areas where high pollution is expected, by restricting traffic on certain roads or resorting to other preventive measures.

A major source of complexity of this task in sensor networks is represented by the geo-distribution of data across multiple locations (as shown in Fig. 1), which violates the typical assumption of independent and identically distributed observations made by machine learning models. The main challenges in this context are the analysis of multivariate data, the necessity to combine information from multiple geo-distributed nodes, and the presence of spatio-temporal autocorrelation described by complex patterns and interactions between sensors. This complexity requires new and sophisticated models that are able to address the aforementioned challenges simultaneously.

While autoregressive models capture temporal autocorrelation, they are typically unable to analyze multivariate data and effectively exploit spatial patterns [1], [2], [3], [4], [5], [6], [7]. Vector autoregression (VAR)-based approaches are able to model spatial dependencies, albeit in a very basic form, by means of coefficients learned solely for a target feature of interest [8], [9].

Manuscript received 30 June 2023; revised 27 December 2023 and 22 February 2024; accepted 2 May 2024. Date of publication 17 May 2024; date of current version 4 September 2024. This work was supported in part by the project FAIR—Future AI Research, Spoke 6—Symbiotic AI funded by the NextGenerationEU under Grant PE00000013 and in part by the EU Project "IMPETUS - Intelligent Management of Processes, Ethics and Technology for Urban Safety" under Grant 883286. *(Corresponding author: Roberto Corizzo.)*

Massimiliano Altieri is with the Department of Computer Science, University of Bari Aldo Moro, 70125 Bari, Italy.

Roberto Corizzo is with the Department of Computer Science, American University, Washington, DC 20016 USA, and also with the Department of Computer Science, University of Bari Aldo Moro, 70125 Bari, Italy (e-mail: rcorizzo@american.edu).

Michelangelo Ceci is with the Department of Computer Science, University of Bari Aldo Moro, 70125 Bari, Italy, also with the Big Data Laboratory, CINI, 00185 Rome, Italy, and also with the Jožef Stefan Institute, 1000 Ljubljana, Slovenia.

Digital Object Identifier 10.1109/TNNLS.2024.3398441

Neural network approaches, including gated recurrent unit (GRU) [10], [11], long short-term memory (LSTM)-based models [12], [13], [14], and other encoder–decoder architectures [15], [16], [17], are able to effectively analyze multivariate data while capturing temporal dependencies, but they only partially address the challenge of combining information from multiple nodes by modeling spatial autocorrelation. In fact, all nodes and features are treated as independent features, therefore ignoring the spatial information in the sensor network. Recently, this issue has been partially addressed by neural network models with custom loss functions [18], methods involving spatio-temporal feature extraction [19], [20], and variants of LSTM such as support vector decomposition (SVD)-LSTM [21] and convolutional neural network (CNN)-LSTM [22], [23], [24], [25], [26], [27], which model local spatial dependencies in data. Another alternative is that of LSTM models with spatial attention [28], [29], [30]. However, these methods do not take into account the geographical structure of nodes, and only have a limited view of neighboring data, bounded by the kernel size, due to the locality characteristic of the convolution operator [31], [32].

A more sophisticated alternative is provided by graph-based recurrent neural networks (RNNs) such as graph convolutional network (GCN)-LSTM [33], [34], [35], [36], [37], [38], which explicitly capture relationships among neighboring nodes. However, one common issue is that spatial patterns are only extracted on low-level features, i.e., the input data representation, before LSTM-based modeling. This aspect may result in a limited ability for deeper layers to exploit more complex spatial correlations, therefore putting more emphasis on temporal information [39], [40], [41], [42], [43].

This article proposes GAP-LSTM, a graph-based autocorrelation-preserving neural network method for geo-distributed forecasting, which addresses all the above-mentioned challenges. Specifically, our method involves a novel GCN-LSTM cell that performs graph convolutions at each timestep, to propagate spatial information throughout the whole time series modeling process, synergically integrating GCN within the LSTM cell. Our cell presents a dual structure. The first part is an LSTM with the addition of GCN layers before the computation of each gate. The second part is a simplified version of the first part that also contains an additional latent memory state: the output gate of the first part of the cell goes directly into the second part, together with the latent memory state of the previous timestep. This novel model architecture allows us to extract and preserve complex spatio-temporal patterns that were hidden in raw data. Moreover, it overcomes the limitation of the commonly adopted GCN-LSTM workflow, where the spatial information is initially extracted by means of a GCN, and an LSTM carries out the forecasting task on the resulting representation. The proposed model architecture is particularly suitable to tackle predictive tasks in many real-world domains, including renewable energy, air pollution, and traffic. These domains are characterized by geo-distributed sensor data with spatio-temporal dependencies that can be exploited to yield more accurate forecasting models.

Another important challenge addressed by our method is that of model interpretability. Indeed, a high model complexity exacerbates the difficulty of gathering insights and explanations about model predictions. This challenge, which is often overlooked by state-of-the-art forecasting methods, is crucial for their adoption as decision support tools in real-world domains such as smart grids. In our method, we leverage the multinode capabilities of our model to generate interpretable attention maps that reveal the most significant nodes and timesteps for the forecasting process, and their relative importance compared to the others in the same sequence and across the whole observation period.

In summary, the main contributions of this work are as follows.

1) A novel model architecture integrating a custom recurrent cell with graph convolution and latent memory states, coupled with attention-based LSTM and 2-D convolution to exploit and preserve spatio-temporal autocorrelation in multivariate and geo-distributed data.
2) An interpretable output visualization that highlights relevant factors that have an impact on predictions, as well as interactions between nodes, supporting domain experts in their decision making.
3) An extensive evaluation with five real-world datasets which shows that GAP-LSTM outperforms state-of-the-art forecasting methods on energy, traffic, and pollution domains. The code to reproduce the experiments is publicly available at https://github.com/m-altieri/GAP-LSTM/.

This article is structured as follows. Section II summarizes related works. Section III describes our proposed method. Section IV describes the experimental settings, and provides a discussion on the results obtained in our experiments. Section V wraps up this article with a summary of the results obtained and outlines relevant directions for future work.

II. Background

A. Autoregressive Models

Autoregressive models for time series forecasting extract future values for a property of interest based on recent values observed for the same property. Popular models include autoregressive integrated moving average (ARIMA) [1], [2] and Prophet [3]. Other works combine autoregressive approaches and evolutionary algorithms to better explore model variants in the search space and select better performing models [4], or combine them with fuzzy or other types of statistical modeling [5], [6], [7]. Among their benefits, it is worthwhile mentioning their ease of use and their determinism, given their statistical nature. On the other hand, their main drawback is that, since they only analyze univariate data, they do not consider exogenous variables. Moreover, they only model temporal autocorrelation, without benefiting from the spatial autocorrelation in data generated by multiple nodes. VAR models [8], [9] partially overcome this limitation, allowing to analyze data from multiple sources.

However, they take into account spatio-temporal autocorrelation only considering the target property subject to the

forecasting task, and attributing equal importance to all nodes. Moreover, similar to ARIMA, they are unable to model complex nonlinear correlations among different values. As a result, vector autoregressive models, only partially address the exploitation of spatial autocorrelation, and do not consider graph-based relationships among nodes.

B. Neural Networks for Temporal and Spatial Data

RNN models, such as LSTM and GRU, overcome some of these limitations by analyzing multivariate data while leveraging nonlinear activation functions, which allow to model and learn more complex correlations among features [10], [11]. LSTM-based modeling has been adopted in a number of applications, including the classification of thermal images of electric motors processed through feature extraction.

Alternative approaches combine a series of multiseasonal decomposition techniques to better capture seasonal patterns in LSTM-based forecasting models [12]. A similar approach is followed in [13], where the forecasting task is carried out using an LSTM model with skip connections complemented with exponential smoothing and ensembling. Other approaches are proposed in [14], [15], [16], and [17], which combine an encoder–decoder model architecture with progressive decomposition capacities for complex time series. However, these methods can only model temporal correlations.

Methods that involve custom spatio-temporal loss functions [18] and feature extraction approaches [19], [20] partially overcome this problem by jointly extracting spatial and temporal dependencies. One popular approach is SVD-LSTM, which performs singular value decomposition and carries out the forecasting task through LSTM networks [21].

A hybrid method of CNN and bi-directional LSTM (Bi-LSTM) models is proposed in [22]. The method learns shared representation features from multivariate time series for air quality data and it is applied to pollution forecasting. CNN-LSTM merges CNN and LSTM models to jointly capture short and long-term spatio-temporal dependencies, as explored in [23], [25], [26], and [27]. A more recent approach in [24] extracts images from sliding windows of time series, and uses saliency as a mixup strategy for data augmentation to train deep models.

The study in [28] proposes a multistep ahead flood forecasting model that features a spatio-temporal attention LSTM, applying attention weights to the hidden layer state of each timestep, while in [29] they use spatio-temporal attention in the encoder, and multiple convolutions in the decoder to extract spatio-temporal features at different resolutions. In [30] spatio-temporal attention is used to predict chlorophyll for the early detection of red tide. More recently, the triangular, variable-specific attentions for long sequence multivariate time series forecasting (Triformer) method [44] proposed a more sophisticated triangular and variable-specific attention mechanism with distinct model parameters for different variables and linear complexity.

Even if these methods support the analysis of multivariate data generated at multiple nodes, an important limitation is that they extract temporal correlations from low-level features in adjacent nodes through an initial modeling step. By doing so, the subsequent operations, which act on top of the extracted high-level features, do not take into account spatial correlations in low-level layers, which are partially unexploited or entirely lost, resulting in a performance degradation on the subsequent downstream task [39], [40], [42], [43]. Some effort has been devoted to mitigate this issue with the introduction of skip connections that propagate spatial information [39], [43] as well as spatial memory cells [40]. Other limitations include the lack of exploitation of graph-based relationships among nodes, and the lack of model interpretability.

C. Graph Neural Networks

A recent and promising thread of research is that of GCNs, which take into account the network structure and combine the contribution of data from multiple sources, by means of a graph convolutional operator [33], [34]. When used in synergy with recurrent networks, GCN can be used to perform time series forecasting tasks.

Some works have focused on the analysis of traffic data. An interesting approach leveraging complete convolutional structures with a limited number of parameters is proposed in [35]. A spatio-temporal graph convolution framework for traffic prediction is proposed in [36], in which multiple graphs are built to explicitly model dynamic correlations among road segments, while RNNs capture temporal correlations for each road segment.

The work in [37] proposes a collaborative graph neural network prediction method involving multiple agents and exploiting dynamic interactions in the system. Interactions are represented as a graph, where edge weights reflect the importance of each predictor. The method has shown successful results on trajectory prediction, online human motion prediction, and online traffic speed prediction.

A different approach is followed in TraverseNet [38], where spatial and temporal dependencies are unified in a non-Euclidean space, and a spatial–temporal graph neural network mines spatial–temporal graphs while exploiting the evolving spatial–temporal dependencies for each node via message traverse mechanisms.

A recent and popular method is graph wavenet (GWN) [45], which exploits an adjacency matrix learned with node embeddings, and a stacked dilated 1-D-convolution component with increasingly growing receptive fields. Another powerful method is regularized graph structure learning (RGSL) [46], which extracts a dense similarity matrix through node embeddings, and learns a sparse graph structure using the regularized graph generation (RGG) method.

Other graph-based approaches were proposed to forecast air quality and electricity data. The method in [47] builds graph models to represent contextual information of nodes and aggregates them via a multigraph fusion module while retaining the importance of each node in the different graphs. A graph neural network with attention as a transfer learning mechanism is proposed in [48], where knowledge from multiple sources is used to improve the learning process in new sources.

The method proposed in [49] addresses the behind-the-meter load and photovoltaic forecasting, modeling the residential units as a spatio-temporal graph where the nodes represent the net load measurements and edges reflect their

TABLE I
COMPARISON OF SOTA METHODS (ROWS) CONSIDERING KEY FEATURES IN GEO-DISTRIBUTED FORECASTING (COLUMNS). FIRST GROUP: GENERAL CLASSES OF APPROACHES. SECOND GROUP: SPECIFIC AND RECENTLY PROPOSED METHODS[1]

Methods	Temporal Autocorrelation (TA)	Spatial Autocorrelation (SA)	Graph-Based Interactions	Interpretability (Int)
Autoregressive	✓	✓		✓
LSTM-based	✓			
Attention-based	✓			✓
CNN-LSTM-based	✓	✓		
Graph-based	✓	✓	✓	
GWN [45]	✓	✓	✓	
MTGNN [51]	✓	✓	✓	✓
Pan et al. [24] [2]	✓	✓		✓
Triformer [44]	✓			✓
ESG [50]	✓	✓	✓	
RGSL [46]	✓	✓	✓	✓
GAP-LSTM	✓	✓	✓	✓

mutual correlation. A graph autoencoder with graph dictionary learning and a deep recurrent structure are used to forecast future generated values at each unit.

The evolutionary multiscale graph (ESG) method [50] leverages a hierarchical graph structure with dilated convolution and updates adjacency matrices to capture scale-specific correlations among time series.

Although all these methods are able to analyze multivariate data from multiple nodes, and both spatial and temporal information is exploited in the learning task, a common pitfall is that they extract spatial and temporal autocorrelation separately. Most frequently, the GCN embeds the spatial component into an encoded vector and the LSTM performs forecasting on top of this representation, which results in the inability to further extract spatial autocorrelation in successive modeling steps. Another common issue of forecasting models is their lack of interpretability, due to the complexity and time-variant nature of the analyzed data. The work in [24] is an interesting attempt in this direction.

A summarized comparison of state-of-the-art forecasting methods is provided in Table I. The comparison emphasizes the novelty of GAP-LSTM with respect to relevant features for geo-distributed forecasting.

III. METHOD

This section is divided into three parts. We first define the problem we tackle in this study. Subsequently, we describe our proposed method in detail, focusing on the contribution provided by each component. Finally, we illustrate the interpretability aspects of the method.

A. Problem Definition

This article addresses the scenario where N geo-distributed nodes generate multivariate observations. Each node is identified by a pair of latitude and longitude coordinates. A graphical overview of the addressed task is shown in Fig. 1. Together, all nodes generate an observation $x_t \in \mathbb{R}^{N\times F}$ for each discrete time point t, F being the number of features. We note that F is the cardinality of the entire feature space consisting of both descriptive (independent) features and target features. In our work, the discrete timeline is split into non-overlapping sequences of length T, each corresponding to a desired unit of analysis, e.g., a single day described by 24 hourly observations. The kth sequence can be defined as follows:

$$s_k = [x_{(k,1)}, x_{(k,2)}, \ldots, x_{(k,T)}] \in \mathbb{R}^{T\times N\times F}. \tag{1}$$

Based on this formulation, it is possible to model data into two data structures:

1) a *sequence tensor* $\mathcal{S} \in \mathbb{R}^{S\times T\times N\times F}$, containing contiguous sequences: $\mathcal{S} = [s_0, s_1, \ldots, s_{S-1}]$, where each sequence s_k contains T chronologically ordered observations;
2) a *graph matrix* $\mathcal{A} \in \mathbb{R}^{N\times N}$, where $\mathcal{A}_{ij}$ is some measure of relationship, or closeness, or correlation, or similarity between nodes i and j.

Given a sequence s_k and a *forecasting horizon* P, the forecasting task consists in approximating the function $f : [x_{(k,1)}, \ldots, x_{(k,T)}] \mapsto [y_{(k,T+1)}, \ldots, y_{(k,T+P)}]$, which returns the next P observations for the target feature of interest, denoted by y.

Without loss of generality, this formalization of the forecasting task also applies to the context of multitarget forecasting, where the goal is to correctly predict a set of target features $F' \leq F$, approximating the ground truth $y \in \mathbb{R}^{P\times N\times F'}$.

In both cases, the aim is to learn a forecasting model Ψ that accurately approximates f. The optimal model Ψ^* is the one that minimizes a generic loss function: $\Psi^* = \arg\min_\Psi ||\Psi(\mathcal{S}) - f(\mathcal{S})||_1$, where $\Psi(\mathcal{S})$ extracts forecasts given the learned model and historical data $[\hat{y}_{(k,T+1)}, \ldots, \hat{y}_{(k,T+P)}]$, and $f(\mathcal{S})$ is the ground truth on historical data for the target variable of interest $[y_{(k,T+1)}, \ldots, y_{(k,T+P)}]$, $\forall k \in \{0, 1, \ldots, S-1\}$.

In addition to the extraction of accurate predictions, our aim is to preserve spatio-temporal autocorrelation throughout the entire learning workflow. Temporal autocorrelation accounts for temporal dependencies between observations at different timesteps [1]. The analysis of temporal autocorrelation allows us to identify and exploit repeating patterns in data characterized by periodic behavior, as commonly observed in time series data on geophysical phenomena (e.g., weather, renewable energy, pollution, traffic). Considering a sequence $s_{k;n}$ collected from a single node n and a desired lag h, temporal autocorrelation can be defined as

$$\mathrm{TA}(s_{k;n}, h) = \frac{\sum_{t=1}^{T-h}(x_{(k;n,t)} - \bar{x}_{(n)})(x_{(k;n,t+h)} - \bar{x}_{(n)})}{\sum_t (x_{(k;n,t)} - \bar{x}_{(n)})^2} \tag{2}$$

where $\bar{x}_{(n)}$ is the average value observed for node n. For all sequences, all nodes, and all possible lags, the average temporal autocorrelation can be computed as the average

$$(1/(S\cdot N\cdot (T-1)))\sum_{k=1}^{S}\sum_{n=1}^{N}\sum_{h=1}^{T-1}\mathrm{TA}(s_{k;n}, h).$$

[1]For SA, a gray tick indicates that a method extracts it without preserving it throughout all the modeling steps. For Int, a gray tick indicates that, even if some aspects of the model could support the interpretation of results (e.g., attention weights), they are not exploited to complement predictions.

[2]This method is not strictly a forecasting method, but it rather provides a general explainability framework for time series analysis.

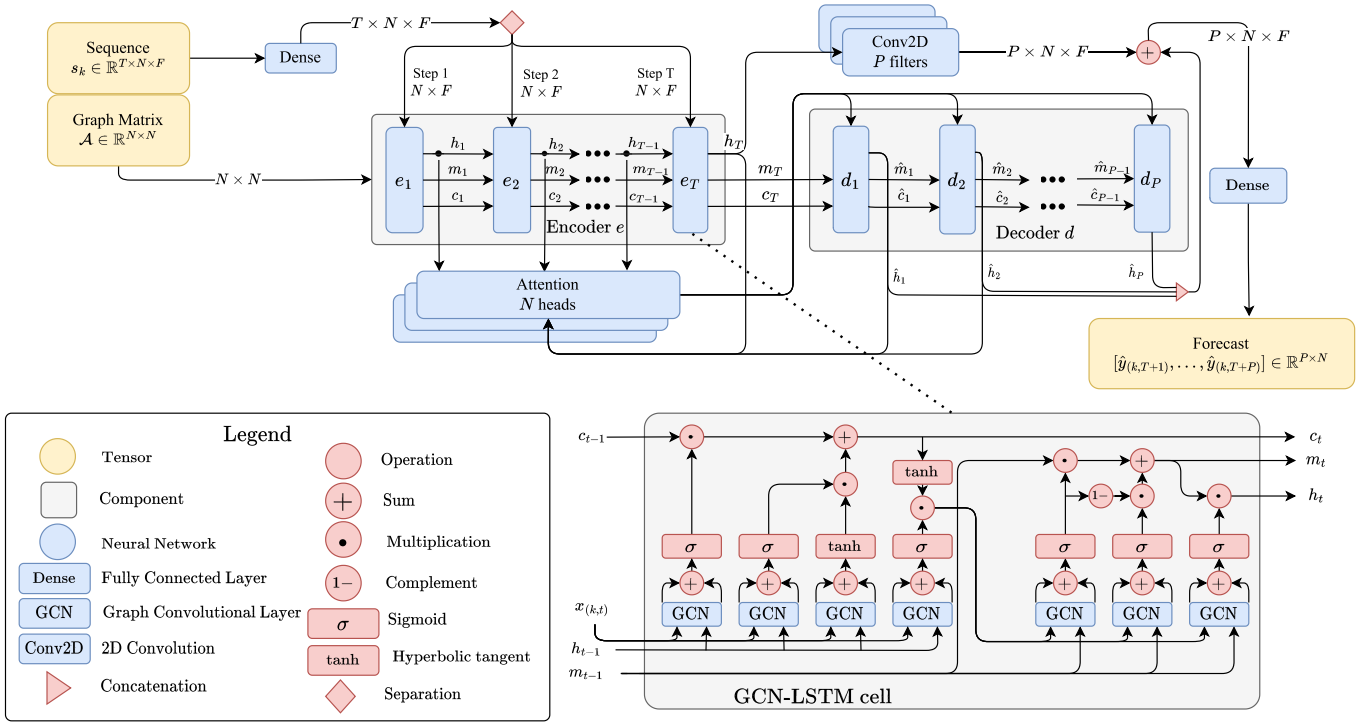

Fig. 2. GAP-LSTM architecture for geo-distributed forecasting. The method leverages graph convolution, attention-based LSTM, 2-D-convolution, and latent memory states to effectively preserve spatio-temporal autocorrelation in the modeling task.

Spatial autocorrelation can be defined as a correlation in a signal among nearby locations in space. A widely adopted way to formalize it is resorting to Moran's I global autocorrelation statistic [52]. Considering a sequence s_k collected from multiple nodes, and the closeness relationships between nodes defined in $\mathcal{A}$, spatial autocorrelation for a single timestep $x_{(k,t)}$ can be defined as

$$\mathrm{SA}(x_{(k,t)}) = \frac{\sum_i \sum_j \mathcal{A}_{ij}(x_{(k;i,t)} - \bar{x}_{(k,t)})(x_{(k;j,t)} - \bar{x}_{(k,t)})}{\sum_i (x_{(k;i,t)} - \bar{x}_{(k,t)})^2} \quad (3)$$

where $\bar{x}_{(k,t)}$ is the average observed value considering all nodes at timestep t. For all sequences, the average spatial autocorrelation can be computed as the average $(1/S \cdot T) \sum_{k=1}^{S} \sum_{t=1}^{T} \mathrm{SA}(x_{(k,t)})$.

B. Proposed Model: GAP-LSTM

The forecasting model proposed in this work involves several deep-learning components. A schematic representation of the method is shown in Fig. 2. In order to simplify the discussion, we first give a general description of the method, based on the sequential information flow, and then we present each component in detail.

1) Method Workflow: The input of the model consists of two tensors: the sequence tensor $\mathcal{S}$ and the graph matrix $\mathcal{A}$, as defined in Section III-A. The model leverages a customized encoder–decoder architecture. The rationale is to use an encoder module e to encode an input sequence s_k to an embedding representation that captures the most relevant information for the task at hand, mitigating possible collinearity and noise in the original data. All encoder and decoder steps are a modified version of the LSTM cell, which includes a graph convolutional layer before each gate,[3] considering the adjacency matrix $\mathcal{A}$ as input, which is used for graph convolution.

Given a sequence s_k and T input steps, we denote with e_t the encoder step t, which processes the observation $x_{(k,t)}$. Each encoder step e_t is a function that takes as input the observation at the current timestep and the outputs of the previous encoder cell, and yields $h_t, m_t, c_t \in \mathbb{R}^{N\times F}$

$$e_t : (x_{(k,t)}, h_{t-1}, m_{t-1}, c_{t-1}) \mapsto (h_t, m_t, c_t) \quad (4)$$

where h_t, m_t, and c_t denote, respectively, hidden state, latent memory state, and cell state for timestep t. Intermediate encoder hidden states are reused in the attention mechanism. While h_t and c_t are standard in LSTM-based cell architectures to capture temporal dependencies, the introduction of the latent memory state m_t is a novel contribution that allows our model to preserve spatio-temporal autocorrelation more effectively than existing LSTM-based models, as discussed later in this section.

The goal of the decoder is to reconstruct the encoded representations into a series of P steps, which will then lead to the prediction. The decoder has the same output shape as the encoder, but may have a different number of steps (in case $T \neq P$). The input of each decoder cell d_p consists of the previous decoder latent memory state $\hat{m}_{p-1}$ and cell state $\hat{c}_{p-1}$ (for d_1, we use the last encoder step's latent memory state m_T and cell state c_T), as well as the weighted sum of encoder hidden states, as determined by the attention mechanism.

[3]The cell architecture is described in detail later in this section.

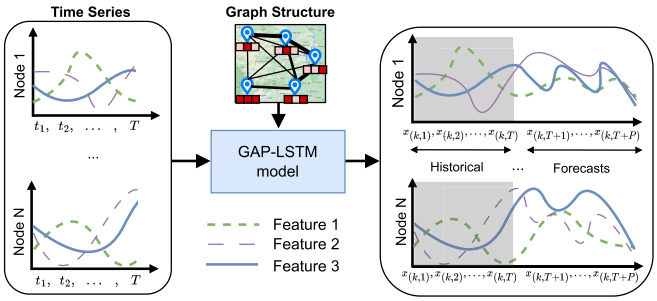

Fig. 3. Graphical workflow of the geo-distributed forecasting task. Historical time series and the graph structure defined by geographical coordinates of the nodes (latitude, longitude) are provided as input to train the GAP-LSTM model, which extracts forecasts for the next p time points.

For a single timestep p, the decoder cell is a mapping

$$d_p : (\{\alpha^{(t)} h_t\}_{t=1}^{T}, \hat{m}_{p-1}, \hat{c}_{p-1}) \mapsto (\hat{h}_p, \hat{m}_p, \hat{c}_p) \tag{5}$$

where $\{\alpha^{(t)} h_t\}$ are the attention weights of the previous timestep. The output of the encoder component h_T is also provided as input to a 2-D-convolutional layer, and then it is added to the decoder hidden states $[\hat{h}_1, \ldots, \hat{h}_P]$.

To extract the final prediction, this tensor then goes through a fully connected layer that reduces the last dimension from F to 1, so that the final output of the model $\Psi(s_k) \in \mathbb{R}^{P \times N}$ contains the value of the relevant target feature for each node at each prediction timestep. A graphical overview of this workflow is shown in Fig. 3 and its high-level pseudo-code is described in Algorithm 1.

Algorithm 1 GAP-LSTM Method Workflow

Input:
$\mathcal{S} = [s_0, s_1, \ldots, s_{S-1}]$ /* seq. tensor (all nodes, hist. data) */
$\mathcal{A}$ /* graph matrix */
$s_k = [x_{(k,1)}, \ldots, x_{(k,T)}]$ /* seq. in $\mathcal{S}$ for all nodes, $k < S$ */
Ψ /* previously trained or randomly initialized model */
Result: Trained model Ψ^* and forecasts $\hat{y}_k$
for $i \in [0, \ldots, k-1]$ **do**
  $\mathcal{L}_i \leftarrow$ training loss, defined as $||\Psi(s_i, \mathcal{A}) - y_i||_1$
  $\Psi^* \leftarrow$ optimize Ψ based on $\mathcal{L}_i$
$\hat{y}_k \leftarrow \Psi^*(s_k, \mathcal{A})$ /* $\hat{y}_k = [\hat{y}_{(k,T+1)}, \ldots, \hat{y}_{(k,T+P)}]$ */
return Ψ^*, $\hat{y}_k$

2) Graph-Based Data Representation: Before introducing the GCN-LSTM cell, we start with describing the graph-based representation for the geo-distributed time series data analyzed in this study. Each node $n \in \{1, 2, \ldots, N\}$ has its own time series describing the features of interest, including the target feature over time.

The nodes and these relationships between nodes can be represented as a graph $G = (V, E)$, where V is the set of all nodes ($|V| = N$), and $E = V \times V$ is the set of edges, or relationships, between nodes. Since the task of interest is to forecast all nodes simultaneously, we can exploit the structural graph information of the data generated at multiple locations to get a better understanding of the whole context, which results in additional spatio-temporal information to support the forecasting task.

An adjacency matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$ is a way to represent relationships between nodes in a compact form. To properly model the intensity of relationships between nodes, in our work, we define entries in the adjacency matrix a_{ij} by exploiting the continuous range of closeness values between all pairs of nodes i, j, i.e., $a_{ij} \in [0, 1]$ represents the closeness relationship between nodes according to their distance. It is calculated as $a_{ij} = 1 - (d(i, j)/\max_{i',j'} d(i', j'))$, where $d(i, j)$ is the geographical distance between nodes i and j. Using this similarity measure, the edges of the graph are undirected, making the adjacency matrix symmetric. In this way, a_{ij} represents how close nodes i and j are relative to the distance between other nodes. If the distance between i and j is 0, their closeness will be 1. In the following, we describe in detail the proposed GCN-LSTM cell occurring in each encoder and decoder step.

3) GCN-LSTM Cell: Our cell is a modified type of the LSTM cell that incorporates graph convolution (GCN), which allows the model to learn spatio-temporal correlations of historical timesteps. The cell has four inputs: the input observation at the current timestep x_t, the previous hidden state h_{t-1}, the previous cell state c_{t-1}, and the previous latent memory state m_{t-1}. The cell also requires the adjacency matrix $\mathcal{A}$ for the GCN computation.

Our method supports the adoption of different forms of graph convolution. Within the scope of this article, we evaluated two graph convolution alternatives, giving place to two variants of our method. The first variant is denoted as GAP-LSTM-Default and adopts the standard GCN implementation in [33], where a graph convolutional layer is defined as: $(X, A) = \text{ReLU}(\hat{A}XW)$, with $\hat{A} = \tilde{D}^{-(1/2)}\tilde{A}\tilde{D}^{-(1/2)}$ for the degree matrix $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $\tilde{A} = A + I_N$ adjacency matrix with added self-loops and $W \in \mathbb{R}^{F \times F'}$ weight matrix. The second variant is denoted as GAP-LSTM-Weighted and employs a slightly different graph convolution implementation, which applies a componentwise weighting process to highlight salient features

$$f(X, A) = \hat{A}XW_{:,[1,\ F]} \odot \sigma(\hat{A}XW_{:,[F+1,\ 2F]})$$

where $W \in \mathbb{R}^{F \times 2F}$, σ denotes the sigmoid activation, and $\odot$ denotes the point-wise (or Hadamard) product.

On the contrary of the typical GCN-LSTM workflow, which first extracts spatial information by means of GCN and then uses LSTM to perform the forecasting task on the resulting representation, we aim to propagate spatio-temporal patterns throughout the entire workflow, synergically integrating GCN within the LSTM cell. This process takes place by means of T graph convolutions, corresponding to T hops in the graph data structure.

The cell is composed of two parts: the first part is similar to a conventional LSTM cell, but with the addition of GCN layers before the computation of each gate. The second part is a simplified version of the LSTM cell, still with the addition of GCN layers before each gate, but also containing an additional latent memory state: the output gate of the first part of the

cell goes directly into the second part, together with the latent memory state. The latent memory state consists of three gates. Conceptually, the first gate acts similar to a forget gate in an LSTM architecture. It filters the information contained in the previous latent memory state by selecting how much latent information from m_{t-1} should be preserved in the current state m_t. The second gate can be seen as an input/update gate that combines old information (m_{t-1}) with new information (output of the left side of the GCN-LSTM cell) by updating the memory state and propagating the information extracted from the first gate. The outputs of the first and second gates are summed and this output is further propagated and filtered through the third gate, which acts as an output gate of the cell, and is in charge of the final output hidden state h_t for the current timestep t. Both hidden states and latent memory states have a size equal to the number of input features F. They are initialized with ones and are updated at each timestep of a given sequence during the inference stage.

An intuitive visual representation of the proposed GCN-LSTM cell is depicted in Fig. 2. Conceptually, the goal of the first part of the cell is to propagate h_{t-1} with the current spatial (GCN) and temporal (LSTM) information. The second part of the cell acquires the output of the first part as well as the latent memory state m_{t-1}, and directs it through GCN and LSTM operations, which further extract spatio-temporal patterns from the latent memory representation. This dual process in our proposed cell has the potential to further emphasize useful spatio-temporal information that was hidden in raw data, and it has been extracted by the first part of the cell. Overall, the GCN-LSTM cell allows our method to extract global spatio-temporal patterns by iteratively combining information in local neighborhoods, based on the closeness relationship among nodes. An illustration of the spatial autocorrelation $\mathrm{SA}(x_{(k,t)})$ measured across the encoder hidden states for different models is shown in Fig. 4. The visualization emphasizes that the proposed GAP-LSTM method presents a better ability to model spatial autocorrelation than other methods due to its cell's ability to jointly propagate temporal and spatial information during all encoding steps.

4) Attention Mechanism: The attention mechanism is used to determine the importance of any encoder step e_t when computing the decoding step d_p, i.e., how much d_p "attends to" e_t, for a prediction step p. Following the attention framework described in [53], we denote with query Q the decoder *thought vector*. The keys $K = \{k_1, k_2, \ldots, k_T\}$ are aligned with the query, to determine how closely related they are to each other. This comparison is performed with a function $a : (k, Q) \mapsto score$, which is usually defined by a trainable neural network, so that each k_t is associated with its own network a_t that learns to properly weigh the corresponding history step t. The outputs of functions $\{a_i\}_1^T$ are normalized with a softmax to get the attention weights $\{\alpha_i\}_1^T$. The values $V = v_1, v_2, \ldots, v_T$ are used to compute the decoder input $\sum_{t=1}^{T} \alpha_t v_t$.

In our method, global multihead attention, also known as soft attention, is adopted [54]. In our formulation, the key and the value are the set of encoder hidden states $K = V = \{h_1, h_2, \ldots, h_T\} \in \mathbb{R}^{T\times N\times F}$.

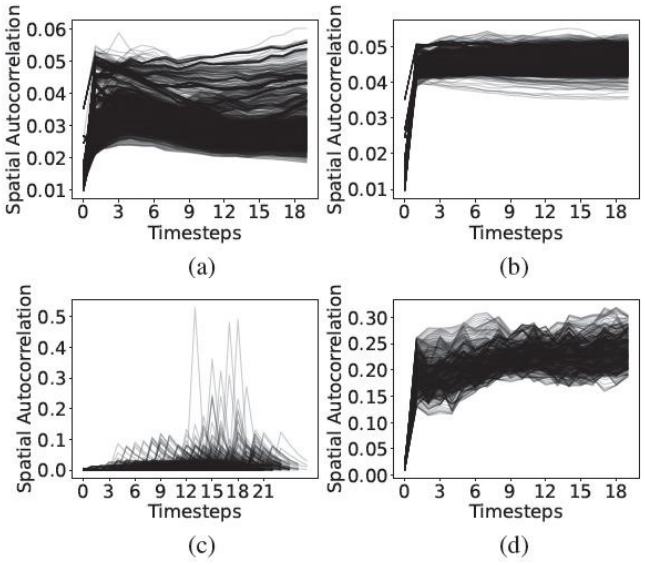

Fig. 4. Spatial autocorrelation according to the global Moran's I statistic for different models. (a) CNN-LSTM, (b) GCN-LSTM, (c) ESG, and (d) GAP-LSTM. The plot shows one line for each training sequence s_k, where the first value is calculated on the original raw input data at timestep 0, and the following values are obtained at each model's encoder hidden states for timesteps $1, 2, \ldots, 19$, corresponding to observations from 2:00 A.M. to 8:00 P.M. in the PV Italy dataset. For ESG, the x-axis labels differ from the other methods and denote intervals rather than timesteps due to the temporal dilation performed by the method.

We denote with $K_n \in \mathbb{R}^{T\times F}$ the key for node n, with $K^{(t)} \in \mathbb{R}^{N\times F}$ the key for history step t, and with $K_n^{(t)} \in \mathbb{R}^F$ the key for node n and history step t. The same notation is used to select specific nodes and history steps from V. We also denote with $Q \in \mathbb{R}^{N\times F}$ the query for the current decoding step d_{p-1}, and with $Q_n \in \mathbb{R}^F$ the query for node n.

The whole attention can be defined as

$$\mathrm{att}(Q, K, V) = \mathrm{Concat}\Big[\mathrm{head}_n(Q_n, K_n, V_n)\Big]_{n\in[1,N]}. \quad (6)$$

Each tensor Q_n, $K_n^{(t)}$, $V_n^{(t)}$, is associated with a learnable weight matrix: $W_{Q;n}$, $W_{K;n}^{(t)}$, and $W_{V;n}^{(t)}$, respectively. Overall, head_n is calculated as

$$\mathrm{head}_n(Q_n, K_n, V_n) = \mathrm{softmax}\Big(Q_n W_{Q;n}\big(K_n W_{K;n}\big)^\top\Big) V_n W_{V;n} \quad (7)$$

where $W_{K;n} = [W_{K;n}^{(t)}]_{t\in[1,T]}$, $W_{V;n} = [W_{V;n}^{(t)}]_{t\in[1,T]}$, and $W_{Q;n}, W_{K;n}^{(t)}, W_{V;n}^{(t)} \in \mathbb{R}^{F\times F}$.

Therefore, the model learns N weight matrices for Q, and $T \cdot N$ weight matrices for K and V, respectively. Keeping the node n fixed, the T matrices for the key and the value can be defined explicitly: for T history steps, we have

$$K_n W_{K;n} = \Big[K_n^{(1)} W_{K;n}^{(1)}, K_n^{(2)} W_{K;n}^{(2)}, \ldots, K_n^{(T)} W_{K;n}^{(T)}\Big] \quad (8)$$

$$V_n W_{V;n} = \Big[V_n^{(1)} W_{V;n}^{(1)}, V_n^{(2)} W_{V;n}^{(2)}, \ldots, V_n^{(T)} W_{V;n}^{(T)}\Big]. \quad (9)$$

The embedded key at each step is a vector $K_n^{(t)} W_{K;n}^{(t)} \in \mathbb{R}^F$. Overall, it is a matrix of size $K_n W_{K;n} \in \mathbb{R}^{T\times F}$, its transpose is $(K_n W_{K;n})^\top \in \mathbb{R}^{F\times T}$, and the attention scores are represented as a vector $Q_n W_{Q;n}(K_n W_{K;n})^\top \in \mathbb{R}^T$, containing a value for each history step.

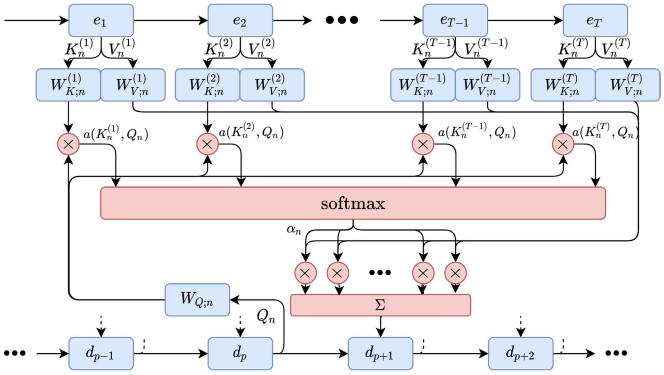

Fig. 5. Global multihead attention mechanism in GAP-LSTM. This diagram represents the computations for a single attention head head_n.

On this vector, a softmax is performed to obtain the attention weights α, which in turn is multiplied by the embedded value $V_n W_{V;n}$, which, like the weighted key, has size $T \times F$. Finally, $\text{head}_n(Q_n, K_n, V_n) \in \mathbb{R}^F$ is a context vector and constitutes the input for the decoder.

Typically, attention mechanisms perform a dot product or extract the embedding of the sole query vector. Instead, in our method, we leverage the embedding of both the query and the key. The advantage is to decouple the encoder information in two parts: the first part h_t, which is used to extract spatio-temporal information from historical timesteps, and the second part $K^{(t)} W_K^{(t)}$, which is useful to model the alignment of each encoder timestep with respect to each decoder state, used to extract predictions. Moreover, in our method, the attention mechanism focuses on each node n individually, in a multihead fashion, learning a different and independent $\text{head}_n(Q_n, K_n, V_n)$, which outputs F features for the given node n.

A graphical representation of our attention mechanism is shown in Fig. 5.

5) 2-D Convolution: A 2-D-convolutional layer is used to modify decoder hidden states $\hat{h}_p$ with a number of filters that extract additional information from the last encoder state h_T. In particular, the convolution computes P filters, one for each prediction timestep, with a kernel of size $(2, 2)$ which is applied to h_T. Different values for kernel size have been tested with preliminary experiments on a variety of domains, and highlighted $(2, 2)$ as the best configuration. However, it can be easily customized to satisfy specific domain characteristics.

The kernel allows us to look at groups of features learned in adjacent nodes and extract useful localized spatial patterns. Specifically, the model will extract, for each timestep p, useful cross-correlations between different features of adjacent nodes to predict that particular timestep. A zero-padding is added and the stride is set to 1, to guarantee that the convolution output keeps the same shape as the input. The resulting P feature maps are stacked together to form a $P \times N \times F$ tensor, which is then added to the decoder hidden states $[\hat{h}_1, \hat{h}_2, \ldots, \hat{h}_P]$ to properly fuse relevant spatial information extracted from the convolution with decoder states.

C. Model Interpretability

This step enhances the GAP-LSTM method with interpretable predictive capabilities. Specifically, our method

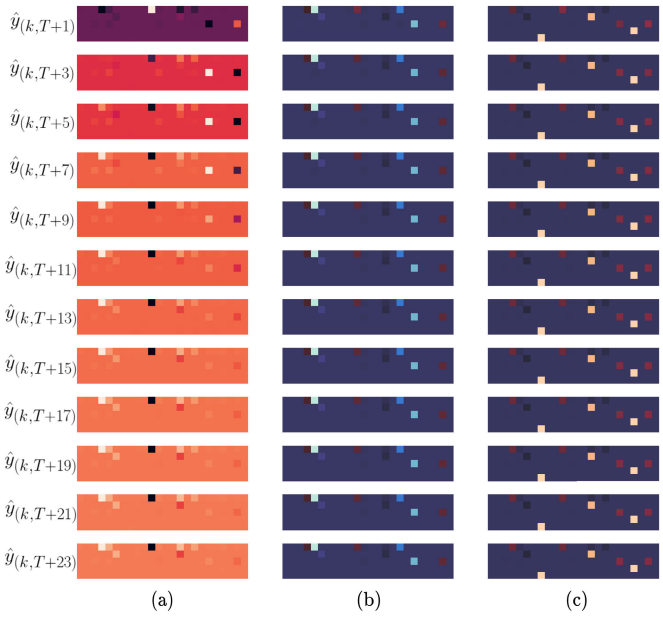

Fig. 6. Interpretable output generated by GAP-LSTM for a given prediction horizon P. The figure illustrates the relevance of encoder hidden states during the decoding stage (prediction) in the attention mechanism by means of p heatmaps. Each heatmap ($N \times T$) shows which nodes ($1, \ldots, N$, 5 in each heatmap) and timesteps ($1, \ldots, T$, 24 in each heatmap) are more relevant (light color) and less relevant (dark color) for the sequence time point subject to prediction. Technically, the figure shows (a) difference between attention weights for each decoding step and the average attention weights for the whole sequence, (b) difference between attention weights for a decoding step and the average attention weights computed for all prediction sequences, and (c) raw attention weights for each decoding step for this sequence. Each row of heatmaps is associated with a single forecasting timestep $\hat{y}_{(k,T+p)}$.

leverages the attention weights α generated at each prediction step p to support the interpretation of the extracted predictions. The rationale is that, for each prediction step, the model generates a different decoding state, which in turn results in different weights α. Combining this information allows us to identify the most relevant encoder hidden states at timesteps $t = 1, 2, \ldots, T$ for the current prediction step p. To accomplish this goal, the attention weights matrix $\alpha \in \mathbb{R}^{N \times T}$ is exploited to generate an insightful visualization that consists of three heatmaps for each p. Let us consider a sequence $s_{\hat{k}}$ and let us denote with $\alpha^{(k)}$ the attention weights for a generic sequence s_k. In Fig. 6, each row $r = 1, \ldots, 12$ shows attention maps computed for decoding step d_{2r-1} during the forecasting of a given sequence. Fig. 6(a) shows the difference between attention weights for each decoding step $\hat{p}$ and average attention weights for this sequence $\alpha^{(\hat{k})}_{:,\hat{p}} - (1/P) \sum_{p=1}^{P} \alpha^{(\hat{k})}_{:,p}$. Fig. 6(b) shows the difference between the attention weights for decoding step $\hat{p}$ and average attention weights computed for all sequences subject to prediction. Let $S_p \subset \{0, 1, \ldots, S-1\}$ be a set of indices referencing sequences subject to prediction, the average attention weight is computed as $\alpha^{(\hat{k})}_{:,\hat{p}} - (1/|S_p|) \sum_{k \in S_p} (1/P) \sum_{p=1}^{P} \alpha^{(k)}_{:,p}$. Fig. 6(c) shows the raw attention weights $\alpha^{(\hat{k})}$ for each decoding step of this sequence.

The difference operations allow us to visually highlight changes in the attention maps corresponding to encoding

steps becoming relevant for a given decoding step in a given sequence. From the visualization in Fig. 6, we can highlight salient patterns in different timesteps where, in each heatmap, rows denote nodes, columns denote encoding steps, and lighter colors in a cell correspond to stronger activations. Let us focus on the attention head for node 1, corresponding to the first row in all heatmaps. Observing the first heatmap at the top in Fig. 6(a) highlights that this attention head was highly aligned to e_{11} for d_1 (as evident from the white pixel), but then gradually shifted its attention toward e_5, e_{15}, e_{17}, and especially e_4 [as evident in the subsequent heatmaps in Fig. 6(a), starting from $\hat{y}_{(k,T+7)}$ to $\hat{y}_{(k,T+23)}$]. This result emphasizes that hidden states h_4, h_5, h_{15}, h_{17} contained particularly useful information for the forecasting of node 1.

IV. Experiments

A. Research Questions

Our experiments are designed to answer the following research questions.

1) *RQ1:* Does GAP-LSTM achieve a higher performance than state-of-the-art methods in the multistep ahead geo-distributed forecasting task?
2) *RQ2:* Is the combination of all components in GAP-LSTM contributing to the achievement of an improved forecasting performance compared to simplified variants of the model?
3) *RQ3:* Can GAP-LSTM provide an effective way for domain experts to interpret the extracted forecasts?

B. Datasets

We perform experiments with the following real-world datasets.

1) **Beijing Air Quality** [55] includes pollutant levels (PM2.5, PM10, SO2, NO2, CO, and O3) in addition to weather features (temperature, pressure, dew point, rain precipitation, and wind speed).
2) **Lightsource** [20] covers solar energy data for the year 2017 from seven plants located in the U.K. Spot values, collected at a time granularity of 1 min, are aggregated hourly.
3) **PEMS SF Weather**[4] describes traffic information in terms of lane occupation, which has been enriched with weather features through the RapidAPI service.
4) **PV Italy** [19] contains hourly observations from 17 solar plants located in Italy, collected from 2:00 A.M. to 8:00 P.M. The time period spans from January 1, 2012 to May 4, 2014.
5) **Wind-NREL** [20] was modeled using the weather research and forecasting (WRF) model. Each plant consists of ten 3-MW turbines (for a total of 30 MW). Hourly aggregated observations range from January 1, 2005 to December 31, 2006.

For all energy datasets (Lightsource, PV Italy, Wind NREL), the following input features are represented: temperature, pressure, wind speed, wind bearing, humidity, dew point, and cloud cover. Lightsource includes additional features for altitude, azimuth, and irradiance. PV Italy includes all features in Lightsource and adds an additional weather summary feature. Lightsource and PV Italy observe a cutoff period between 9:00 P.M. and 2:00 A.M. due to the absence of irradiance at that time, i.e., plants are not operational and no observations are recorded during that time frame. For all datasets, the latitude and longitude coordinates of nodes are used to extract the graph matrix $\mathcal{A}$.

[4]http://pems.dot.ca.gov

TABLE II
Datasets Analyzed in Our Experiments

Dataset	Domain	Time Frame	Horizon P	Nodes N	Features F
Beijing Air Quality	Pollution	≈4 years	6	12	11
Lightsource	Energy	1 year	19	7	11
PEMS SF Weather	Traffic	≈1.2 years	6	163	16
PV Italy	Energy	≈2.5 years	19	17	12
Wind NREL	Energy	2 years	24	5	8

Fig. 7. Real photographs for the different datasets considered in our experiments: solar plant arrays in the south of Italy (PV Italy), 3-MW wind turbines (Wind NREL), traffic lanes in the San Francisco Bay Area (PEMS SF Weather), and Beijing's Municipal Environmental Monitoring Center (Beijing Air Quality).

A summary of our analyzed datasets is shown in Table II. A set of representative photos for the considered datasets is shown in Fig. 7.

C. Experimental Setup

We perform a data pre-processing phase consisting in the selection of significant features, conversion of all features to real values (including one-hot encoding of categorical features), min-max normalization of all features in the [0, 1] range, and computation of pair-wise node closeness according to their geographical location [18]. Considering the sequential nature of data analyzed in our study, the model receives the sequences (each representing 24 h) one by one (every day) in chronological order. Models are trained considering all historical data sequences $s_0, s_1, \ldots, s_{S-1}$. Given the current sequence s_k, the model extracts predictions according to the forecasting horizon P (see Table II). For all experiments, models are optimized through Stochastic Gradient Descent using the Adam optimizer for 50 epochs, and mean absolute error (MAE) as the loss function. Fine-tuning takes place via grid search using validation data (1% of available sequences in each dataset) considering the following sets of hyperparameter values: batch size: $\{2^i\}_{i\in\{2,3,4\}}$; learning rate (LR): $\{10^j\}_{j\in\{-3,-4,-5\}}$.

Each method featured in our experiments is optimized separately for each dataset considering the aforementioned search space for hyperparameters. Once the optimal configuration on validation sequences for a given method is found, it is selected

TABLE III
EXPERIMENTAL RESULTS FOR ALL THE METHODS AND DATASETS UNDER EVALUATION IN TERMS OF MAE, SMAPE, AND RMSE, WITH THE RESPECTIVE VARIANCE IN PARENTHESIS. THE BEST-PERFORMING METHOD FOR EACH DATASET ACCORDING TO THE RMSE METRIC IS MARKED IN BOLD

Method	Beijing Air Quality			Lightsource			PEMS-SF Weather		
	MAE	SMAPE	**RMSE**	MAE	SMAPE	**RMSE**	MAE	SMAPE	**RMSE**
ARIMA [1]	0.0477 (.01)	32.2 (.03)	0.0598 (.01)	0.1248 (.02)	57.0 (.02)	0.1666 (.01)	—	—	—
LSTM [12], [13]	0.0744 (.01)	47.0 (.07)	0.0843 (.01)	0.1144 (.02)	51.1 (.04)	0.1621 (.01)	0.0308 (.01)	31.9 (.03)	0.0440 (.01)
GRU [11]	0.0684 (.01)	43.4 (.07)	0.0779 (.01)	0.0811 (.02)	47.0 (.02)	0.1272 (.01)	0.0261 (.01)	26.9 (.03)	0.0392 (.01)
Bi-LSTM [22]	0.1007 (.01)	55.9 (.08)	0.1118 (.01)	0.0834 (.02)	47.0 (.02)	0.1288 (.01)	0.0282 (.01)	27.5 (.02)	0.0418 (.01)
Attention-LSTM [28]–[30]	0.0755 (.01)	41.5 (.06)	0.0882 (.01)	0.0907 (.02)	48.9 (.02)	0.1398 (.01)	0.0251 (.01)	26.0 (.02)	0.0390 (.01)
CNN-LSTM [22]–[27]	0.0490 (.01)	33.7 (.03)	0.0587 (.01)	0.1133 (.02)	51.4 (.04)	0.1582 (.01)	0.0295 (.01)	28.8 (.03)	0.0436 (.01)
SVD-LSTM [21]	0.0595 (.01)	47.1 (.04)	0.0710 (.01)	0.1168 (.02)	51.4 (.04)	0.1623 (.01)	0.0289 (.01)	30.1 (.03)	0.0422 (.01)
GCN-LSTM [33]–[38]	0.0503 (.01)	36.3 (.02)	0.0603 (.01)	0.1100 (.02)	53.3 (.02)	0.1561 (.01)	0.0336 (.01)	33.1 (.03)	0.0460 (.01)
GWN [45]	0.0411 (.01)	32.8 (.04)	0.0509 (.01)	0.0821 (.02)	47.3 (.03)	0.1256 (.01)	0.0275 (.01)	29.1 (.02)	0.0408 (.01)
MTGNN [51]	0.0385 (.01)	29.8 (.03)	0.0482 (.01)	0.0776 (.02)	45.0 (.03)	0.1252 (.01)	0.0274 (.01)	28.1 (.02)	0.0410 (.01)
Triformer [44]	0.0446 (.01)	31.5 (.03)	0.0559 (.01)	0.0785 (.02)	44.3 (.03)	0.1245 (.01)	0.0264 (.01)	26.4 (.02)	0.0397 (.01)
ESG [50]	0.0312 (.01)	25.0 (.03)	**0.0412** (.01)	0.0786 (.02)	45.8 (.03)	0.1257 (.01)	0.0267 (.01)	28.5 (.02)	0.0405 (.01)
RGSL [46]	0.0432 (.01)	31.2 (.04)	0.0522 (.01)	0.0809 (.02)	47.8 (.03)	0.1223 (.01)	0.0281 (.01)	27.9 (.02)	0.0410 (.01)
GAP-LSTM-Default	0.0366 (.01)	28.8 (.03)	0.0456 (.01)	0.1084 (.02)	52.8 (.03)	0.1570 (.01)	0.0272 (.01)	28.1 (.03)	0.0409 (.01)
GAP-LSTM-Weighted	0.0387 (.01)	29.1 (.03)	0.0494 (.01)	0.0763 (.02)	45.3 (.03)	**0.1213** (.01)	0.0251 (.01)	26.4 (.03)	**0.0387** (.01)
GAP-LSTM-GAT	0.0497 (.01)	34.3 (.04)	0.0589 (.01)	0.1015 (.02)	52.4 (.02)	0.1519 (.01)	0.0286 (.01)	27.4 (.02)	0.0416 (.01)

Method	PV Italy			Wind NREL		
	MAE	SMAPE	**RMSE**	MAE	SMAPE	**RMSE**
ARIMA [1]	0.1229 (.00)	64.5 (.02)	0.1677 (.01)	0.2297 (.04)	37.9 (.05)	0.2804 (.02)
LSTM [12], [13]	0.0572 (.01)	48.6 (.02)	0.1051 (.01)	0.2740 (.07)	43.8 (.05)	0.3280 (.04)
GRU [11]	0.1233 (.02)	66.1 (.02)	0.1705 (.01)	0.2792 (.06)	41.6 (.06)	0.3329 (.03)
Bi-LSTM [22]	0.0626 (.02)	50.7 (.03)	0.1099 (.01)	0.2705 (.07)	41.8 (.05)	0.3249 (.04)
Attention-LSTM [28]–[30]	0.0586 (.02)	48.9 (.02)	0.1064 (.01)	0.2662 (.07)	42.1 (.05)	0.3201 (.04)
CNN-LSTM [22]–[27]	0.0932 (.02)	53.9 (.03)	0.1417 (.01)	0.2408 (.06)	41.2 (.05)	0.2964 (.03)
SVD-LSTM [21]	0.0591 (.01)	49.7 (.02)	0.1027 (.01)	0.2367 (.06)	39.1 (.05)	0.2925 (.03)
GCN-LSTM [33]–[38]	0.0925 (.02)	57.0 (.02)	0.1374 (.01)	0.3387 (.07)	52.2 (.02)	0.3834 (.04)
GWN [45]	0.0659 (.01)	50.3 (.02)	0.1057 (.01)	0.2416 (.04)	38.3 (.05)	0.2852 (.02)
MTGNN [51]	0.0641 (.01)	49.9 (.02)	0.1055 (.01)	0.2305 (.05)	38.3 (.05)	0.2784 (.02)
Triformer [44]	0.0573 (.01)	48.5 (.02)	0.1039 (.01)	0.2649 (.06)	41.1 (.05)	0.3108 (.03)
ESG [50]	0.0592 (.02)	49.6 (.02)	0.1055 (.01)	0.2310 (.05)	38.9 (.05)	0.2786 (.02)
RGSL [46]	0.0644 (.01)	51.5 (.02)	0.1031 (.01)	0.2557 (.04)	39.6 (.05)	0.2947 (.02)
GAP-LSTM-Default	0.0589 (.01)	50.4 (.02)	**0.1024** (.01)	0.2270 (.05)	38.8 (.05)	**0.2762** (.02)
GAP-LSTM-Weighted	0.0916 (.02)	53.7 (.03)	0.1406 (.01)	0.2692 (.06)	43.0 (.06)	0.3219 (.03)
GAP-LSTM-GAT	0.0676 (.01)	52.0 (.02)	0.1069 (.01)	0.2496 (.03)	38.6 (.05)	0.2838 (.02)

for the actual experiments carried out on prediction sequences (testing set). In order to evaluate the models, 10% of the available 24-h sequences are randomly sampled from each dataset for testing purposes. Once sampled, testing dates are fixed and the same dates are used to fairly compare performance across different methods. Moreover, to simulate the outcome of multiple independent executions of the method while ensuring that only historical data from previous timesteps is being used, the evaluation (training on data prior to the prediction sequence and extraction of forecasts for the prediction sequence) is repeated for each prediction sequence.

For a given ground-truth sequence and its respective model forecast, the evaluation of the forecasting accuracy of each model is performed using the following metrics.

1) *MAE:*

$$\text{MAE} = \frac{1}{S}\sum_{k=0}^{S-1}\left[\frac{1}{P}\sum_{p=1,\ldots,P}|\hat{y}_{(k,T+p)} - y_{(k,T+p)}|\right].$$

2) *Root-mean-square error (RMSE):*

$$\text{RMSE} = \frac{1}{S}\sum_{k=0}^{S-1}\left[\sqrt{\frac{1}{P}\sum_{p=1,\ldots,P}(\hat{y}_{(k,T+p)} - y_{(k,T+p)})^2}\right].$$

3) *Symmetric mean absolute percentage error (SMAPE):*

$$\text{SMAPE} = \frac{1}{S}\sum_{k=0}^{S-1}\left[\frac{100}{P}\sum_{p=1}^{P}\frac{|\hat{y}_{(k,T+p)} - y_{(k,T+p)}|}{|\hat{y}_{(k,T+p)}| + |y_{(k,T+p)}|}\right].$$

D. Results

Experimental results for all methods with all datasets are reported in Table III. Results are averaged over all test executions (10% of sequences).[5] The results highlight that at least one variant of the method (GAP-LSTM-Weighted or GAP-LSTM-Default) achieves the best forecasting accuracy in terms of average RMSE, for all the analyzed datasets with the exception of Beijing Air Quality, where our method provides the second-best results. Particularly, GAP-LSTM-Weighted performs better on the Lightsource and PEMS-SF Weather datasets, possibly due to the presence of highly correlated clusters of nodes operating similarly, as well as the high similarity and regularity of solar energy prediction and traffic curves in these datasets. For the other three datasets, the basic weighting provided by the closeness relationship

[5]The materials to replicate our experiments are available at the following repository: https://github.com/m-altieri/GAP-LSTM/.

TABLE IV
STATISTICAL ANALYSIS WITH WILCOXON SIGNED RANK TESTS COMPARING ALL PAIRWISE COMBINATIONS OF METHODS WITH GAP-LSTM-DEFAULT. THE +(−) SIGN DENOTES THAT GAP-LSTM OUTPERFORMS (DOES NOT OUTPERFORM) THE COMPETITOR BASED ON ITS AVERAGE SINGLE-DATASET PERFORMANCE (RMSE). BOLD TEXT DENOTES COMPARISONS THAT ARE STATISTICALLY SIGNIFICANT (p-VALUE < 5.0E-02)

Competitors	Beijing	Lightsource	PEMS	PV Italy	Wind NREL
ARIMA	**3.2E-07 +**	**3.3E-02 +**	**3.8E-09 +**	**9.6E-14 +**	2.23E-1 +
LSTM	**2.2E-16 +**	4.5E-01 +	**3.8E-02 +**	3.6E-01 +	**1.2E-03 +**
GRU	**6.0E-16 +**	**1.1E-02 -**	1.7E-01 -	**4.4E-11 +**	**1.3E-02 +**
Bi-LSTM	**2.4E-23 +**	**9.2E-03 -**	2.2E-01 +	1.6E-01 +	**4.2E-03 +**
Attention-LSTM	**1.1E-17 +**	1.6E-01 -	2.4E-01 -	1.6E-01 +	**1.2E-03 +**
CNN-LSTM	**2.3E-09 +**	4.3E-01 +	6.9E-02 +	**8.6E-10 +**	8.2E-02 +
SVD-LSTM	**3.4E-12 +**	4.7E-01 +	3.4E-01 +	3.5E-01 +	**3.3E-02 +**
GCN-LSTM	**2.0E-10 +**	3.6E-01 -	4.1E-02 +	**7.7E-10 +**	**6.4E-06 +**
GWN	**3.5E-04 +**	**5.2E-03 -**	4.5E-01 -	**8.3E-02 +**	2.9E-01 +
MTGNN	**4.2E-02 +**	**1.3E-02 -**	1.9E-01 +	2.0E-01 +	4.1E-01 +
Triformer	**1.0E-09 +**	**1.9E-04 -**	2.7E-01 -	4.9E-01 +	**4.8E-03 +**
ESG	**1.3E-03 -**	**1.1E-02 -**	4.8E-01 -	2.9E-01 +	4.4E-01 +
RGSL	**1.0E-02 +**	**2.4E-05 -**	3.7E-01 +	4.5E-01 +	**2.9E-02 +**

Comparisons	+ Signif.	+ Non Signif.	- Non Signif.	- Signif.
65	28	22	7	8

among nodes is apparently enough to model the actual spatial dependencies. Finally, we observe that the GAT variant or our method, GAP-LSTM-GAT, is not able to achieve the same performance of the other two variants. In summary, results show that GAP-LSTM generally achieves the highest forecasting performance across all methods and all datasets in terms of RMSE. To validate the statistical significance of our results, we perform Wilcoxon Signed Rank tests to all pairwise combinations of methods across multiple executions with all datasets. The results in Table IV highlight that GAP-LSTM-Default is the most robust variant of our method, outperforming competitors in 50 out of 65 configurations, 28 of which are statistically significant. GAP-LSTM-Weighted outperforms other approaches 44 times, 20 of which are statistically significant. Among all competitors, we observe that ESG performs similar to GAP-LSTM-Default. However, it is noteworthy that ESG is not able to exploit the possibility of fully catching spatio-temporal autocorrelation (as evident in Fig. 4) and, in addition, does not generate interpretable results. Moreover, it generally shows a worse performance with large prediction horizons (e.g., Wind NREL, PV Italy), with the exception of Lightsource, where ESG achieves a slightly worse performance than GAP-LSTM-Weighted (**RQ1**).

A different view of results is obtained considering model calibration by analyzing the number of times each model's predictions overestimate or underestimate target values. This analysis reveals interesting border cases of imbalanced predictions for specific models and datasets. We observe that Attention-LSTM overestimates target values in the 68.36% of the cases with the Beijing Air Quality dataset. CNN-LSTM overestimates 65.75% with the Lightsource dataset. On PV Italy, GRU underestimates target values in 65.55% of the cases, whereas CNN-LSTM overestimates them in 67.59%. Finally, we observe that GWN underestimates target values in 62.89% of the cases with Wind NREL.

Among the competitors considered in our study, autoregressive models (ARIMA) present an unsatisfactory performance, since they do not fully exploit the multivariate nature of data, and do not combine the spatial information residing in the multinode graph structure. We note that ARIMA was not evaluated on PEMS-SF Weather due to the unfeasible computational cost observed with this dataset. As for neural network-based methods, they significantly outperform ARIMA, but generally present a substandard performance. This result is justified by the fact that temporal-focused methods (LSTM, GRU, Bi-LSTM, Attention-LSTM) do not exploit spatial dependencies, whereas spatio-temporal methods (CNN-LSTM and GCN-LSTM) partially exploit them, as discussed in Section II. Shifting the focus to more recent baselines (GWN, multivariate time series forecasting with graph neural network (MTGNN), Triformer, ESG, RGSL), we can observe that they generally achieve very competitive results, significantly outperforming the aforementioned competitor methods, but appear suboptimal when compared to GAP-LSTM, with the exception of Beijing Air Quality dataset, where ESG outperforms all other approaches. A possible reason is that pollution predictions are more accurate with multiscale temporal modeling rather than spatio-temporal autocorrelation preserving modeling. However, with the exception of ESG, GAP-LSTM outperforms all other methods with this dataset. The superiority of GAP-LSTM on all other datasets is due to its competitors' inability to model spatial dependencies (Triformer), or their inability to preserve them throughout all modeling steps (MTGNN, ESG). The other methods (GWN, RGSL) effectively model and preserve spatial dependencies, but their scope is likely limited to one or few types of spatial interactions without considering, for instance, cross-correlations in the embedding space, and appear penalized when such characteristics are naturally present in the data.

We conducted an ablation study to verify that all components of GAP-LSTM contribute to achieving a higher forecasting performance, and report its results in Table V. In this analysis, we run experiments with different variants of the model where the 2-D-convolutional layer, the attention mechanism, and the second part of the GCN-LSTM cell are mutually excluded. When one of the components is deactivated, the variant GAP-LSTM-NoMemoryState yields the second-best performance, followed by GAP-LSTM-NoAttention and GAP-LSTM-NoConv. We can also observe that deactivating any of the components always leads to a performance degradation ranging from 2.0% to 38.7% with respect to the best variant of GAP-LSTM in terms of RMSE.

The experimental results obtained for all variants of the model are reported at the bottom of Table III, and highlight that all components provide a positive contribution in terms of model accuracy. As a result, model variants where all components are active (GAP-LSTM-Weighted and GAP-LSTM-Default) achieve the best overall performance. From a qualitative viewpoint, Fig. 8 shows multinode prediction (red) and measured (blue) curves (one curve per node) for the target variable of each dataset and for a selected day.[6]

[6]We do not report PEMS SF Weather curves due to the very large number of nodes which results in a cluttered visualization.

TABLE V
ABLATION STUDY. THE COMPLETE MODEL IS COMPARED WITH THREE VARIANTS WHERE THE 2-D CONVOLUTION, THE ATTENTION MECHANISM, AND THE SECOND PART OF THE GCN-LSTM CELL ARE MUTUALLY EXCLUDED. THE BEST-PERFORMING METHOD FOR EACH DATASET ACCORDING TO THE RMSE METRIC IS MARKED IN BOLD. THE DEGRADATION COLUMN MEASURES ERROR INCREASE AFTER DEACTIVATING EACH COMPONENT

Beijing Air Quality				
Model	MAE	SMAPE	**RMSE**	Degradation
GAP-LSTM-Default	0.0366 (.01)	28.8 (.03)	**0.0456** (.01)	—
GAP-LSTM-Weighted	0.0387 (.01)	29.1 (.03)	0.0494 (.01)	—
GAP-LSTM-GAT	0.0497 (.01)	34.3 (.04)	0.0589 (.01)	—
GAP-LSTM-NoConv	0.0395 (.01)	31.2 (.04)	0.0487 (.01)	6.8%
GAP-LSTM-NoAttention	0.0378 (.01)	30.8 (.04)	0.0473 (.01)	3.7%
GAP-LSTM-NoMemoryState	0.0373 (.01)	29.4 (.03)	0.0467 (.01)	2.4%
Lightsource				
Model	MAE	SMAPE	**RMSE**	Degradation
GAP-LSTM-Default	0.1084 (.02)	52.8 (.03)	0.1570 (.01)	—
GAP-LSTM-Weighted	0.0763 (.02)	45.3 (.03)	**0.1213** (.01)	—
GAP-LSTM-GAT	0.1015 (.02)	52.4 (.02)	0.1519 (.01)	—
GAP-LSTM-NoConv	0.1172 (.02)	53.2 (.04)	0.1683 (.01)	38.7%
GAP-LSTM-NoAttention	0.1119 (.02)	53.5 (.03)	0.1611 (.01)	32.8%
GAP-LSTM-NoMemoryState	0.0989 (.02)	50.9 (.04)	0.1427 (.01)	17.6%
PEMS-SF Weather				
Model	MAE	SMAPE	**RMSE**	Degradation
GAP-LSTM-Default	0.0272 (.01)	28.1 (.03)	0.0409 (.01)	—
GAP-LSTM-Weighted	0.0251 (.01)	26.4 (.03)	**0.0387** (.01)	—
GAP-LSTM-GAT	0.0286 (.01)	27.7 (.02)	0.0416 (.01)	—
GAP-LSTM-NoConv	0.0272 (.01)	27.1 (.02)	0.0408 (.01)	5.4%
GAP-LSTM-NoAttention	0.0293 (.01)	30.8 (.03)	0.0422 (.01)	9.0%
GAP-LSTM-NoMemoryState	0.0270 (.01)	28.7 (.02)	0.0404 (.01)	4.4%
PV Italy				
Model	MAE	SMAPE	**RMSE**	Degradation
GAP-LSTM-Default	0.0589 (.01)	50.4 (.02)	**0.1024** (.01)	—
GAP-LSTM-Weighted	0.0916 (.02)	53.7 (.03)	0.1406 (.01)	—
GAP-LSTM-GAT	0.0676 (.01)	52.0 (.02)	0.1069 (.01)	—
GAP-LSTM-NoConv	0.0622 (.01)	51.1 (.02)	0.1057 (.01)	3.9%
GAP-LSTM-NoAttention	0.0599 (.01)	50.0 (.02)	0.1042 (.01)	2.0%
GAP-LSTM-NoMemoryState	0.0614 (.01)	50.7 (.02)	0.1061 (.01)	3.9%
Wind NREL				
Model	MAE	SMAPE	**RMSE**	Degradation
GAP-LSTM-Default	0.2270 (.05)	38.8 (.05)	**0.2762** (.02)	—
GAP-LSTM-Weighted	0.2692 (.06)	43.0 (.06)	0.3219 (.03)	—
GAP-LSTM-GAT	0.2496 (.03)	38.6 (.05)	0.2838 (.02)	—
GAP-LSTM-NoConv	0.3003 (.07)	43.0 (.07)	0.3519 (.03)	27.5%
GAP-LSTM-NoAttention	0.2624 (.06)	41.6 (.05)	0.3128 (.03)	13.4%
GAP-LSTM-NoMemoryState	0.2470 (.05)	41.3 (.05)	0.2955 (.02)	7.2%

Overall, GAP-LSTM outperforms state-of-the-art methods, highlighting that the synergic work of all components contributes to the extraction of the most accurate predictions in the final model (**RQ2**). Specifically, the 2-D-convolutional layer fruitfully extracts spatial patterns from the last step of the encoder output using information from all nodes that are useful for the decoder. In addition, the attention mechanism determines the importance of previous timesteps in the input sequence during the generation of each prediction step performed by the decoder. Finally, the latent memory state component in the GCN-LSTM cell enhances spatio-temporal patterns residing in the embedded representation of each timestep of the sequence with a latent representation of the previous timestep.

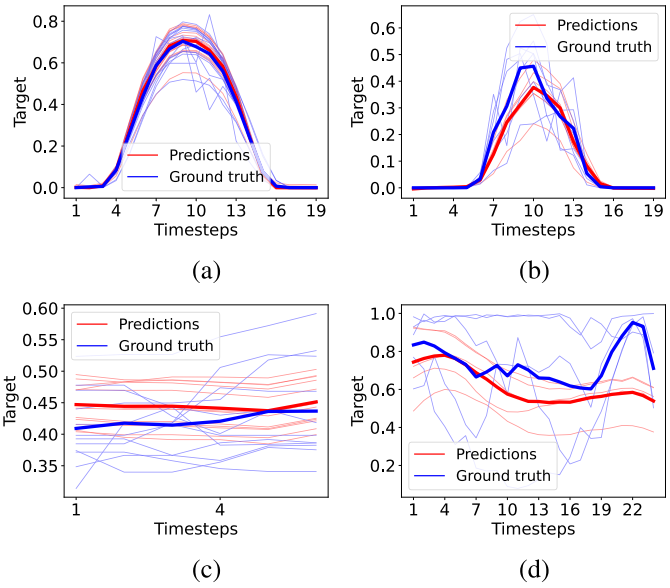

Fig. 8. Predicted (red) versus actual (blue) curves for different nodes on one selected test day in different datasets. (a) PV Italy. (b) Lightsource. (c) Beijing Air Quality. (d) Wind NREL. Each line is the time series for a single node. The bolded line represents the average across all nodes.

Focusing on model interpretability, we leverage the attention weights α generated at each prediction step p to support the interpretation of the extracted predictions. The rationale is that, for each prediction step, the model generates a different decoding state, which in turn results in different weights α. Combining this information allows to identify the most relevant encoder hidden states at timesteps $t = 1, 2, \ldots, T$ for the current prediction step p. Visualizations in Fig. 6 provide domain experts with this information, supporting them in visually understanding the most relevant factors influencing model predictions. This visualization demonstrates our method's ability to extract a succinct and qualitative interpretation of the forecast values, providing an effective way for domain experts to understand the extracted forecasts and support their decision-making process (**RQ3**).

Another way to gather insights about model's predictions is to demonstrate the effective exploitation of spatio-temporal autocorrelation provided by the different components of GAP-LSTM.

To gain deeper insights into the decoder representations, we adopt t-distributed stochastic neighbor embedding (t-SNE), which effectively extracts 2-D visualizations from the $N \times F$ decoder hidden states for a given prediction hour. In Fig. 9(a)–(e), we show four visualizations corresponding to four equally spaced hours. Points correspond to nodes of the Lightsource dataset, whereas the X- and Y-axes correspond to the compressed representation space extracted by t-SNE.

The results highlight that the embedding representation learned by our model preserves spatio-temporal correlations among different nodes, which are evident at different timesteps [see Fig. 9(a)–(d)] during the predictive stage. Similar nodes appear naturally clustered by the method based on their behavior, and resemble the closeness relationship defined by their physical location in the sensor network, as depicted in

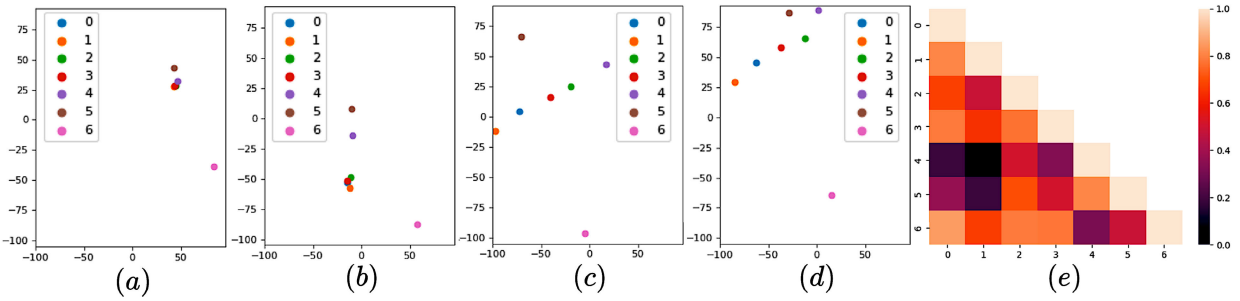

Fig. 9. Visualization of the decoder hidden states extracted with t-SNE for a randomly sampled predicted sequence of the Lightsource dataset at different hours. (a) 4 A.M., (b) 9 A.M., (c) 2 P.M., and (d) 7 P.M., and (e) corresponding closeness heatmap for all nodes. Distances between nodes in (a)–(d) resemble physical closeness relationships among nodes in (e).

TABLE VI

EXECUTION TIMES FOR THE WHOLE TRAINING PROCESS FOR ALL THE METHODS AND DATASETS. BAQ DENOTES THE BEIJING AIR QUALITY DATASET

Method	Execution Times (s)				
	BAQ	Lightsource	PEMS-SF	Weather	PV Italy	Wind NREL
ARIMA [1]	6,654	4,374	—	77,758	40,163
LSTM [12], [13]	41	45	28	341	547
GRU [11]	38	71	130	92	589
Bi-LSTM [22]	52	219	96	159	309
Att-LSTM [28]–[30]	309	157	87	296	290
CNN-LSTM [22]–[27]	355	42	42	104	221
SVD-LSTM [21]	145	185	255	1,261	775
GCN-LSTM [33]–[38]	131	92	120	204	220
GWN [45]	2,133	498	903	1,181	997
MTGNN [51]	2,032	585	822	1,051	1,287
Triformer [44]	1,705	391	635	1,163	767
ESG [50]	3,067	674	1,457	2,171	1,654
RGSL [46]	3,320	1,632	1,908	3,654	4,626
GAP-LSTM	1,987	710	9,659	6,826	11,626
GAP-LSTM-NoConv	1,784	683	9,720	5,515	8,429
GAP-LSTM-NoAttention	681	192	499	760	3,547
GAP-LSTM-NoMS	1,759	934	9,061	5,753	10,203

TABLE VII

EMPIRICAL ANALYSIS ON GAP-LSTM SPACE AND TIME COMPLEXITY, INCLUDING THE AMOUNT OF MEMORY, TIME, RELATIVE TIME (WITH RESPECT TO THE SMALLEST EXPERIMENT WITH FIVE NODES), AND TIME-PER-NODE (TPN) WITH AN INCREASING NUMBER OF NODES IN THE GRAPH

Nodes	**Memory**			**Time**		
	Abs. (MB)	Rel.	Per-node	Abs. (s/epoch)	Rel.	Per-node
5	1552	1.00	310.4	12	1.00	2.40
10	1560	1.01	156.0	19	1.58	1.90
15	1564	1.01	104.3	29	2.41	1.93
20	1640	1.06	82.0	36	3.00	1.80
25	1644	1.06	65.8	48	4.00	1.92
30	1776	1.14	59.2	63	5.25	2.10
50	2032	1.31	40.6	82	6.83	1.64

Fig. 9(e): lighter colors correspond to pairs of nodes with higher values of closeness. For instance, nodes 2 and 3 present a value close to 0.8 in Fig. 9(e), and appear systematically close to each other (red and green points) in Fig. 9(a)–(d) (**RQ3**).

E. Time Complexity

To evaluate the computational cost of all methods, we computed their execution time as the ratio between the total training time of each method and the number of prediction sequences for each dataset (number of executions). The result is then averaged across all datasets. In the following, we discuss the results for a subset of representative methods from different categories, which yield an accurate performance.

Experiments are run on a workstation equipped with an Intel Xeon W-2145 (3.7 GHz) CPU, 64 GB of RAM (DDR4-2666), 512-GB SSD drive, and an NVIDIA RTX 4090 GPU. Execution times for all methods and datasets are reported in Table VI. We observe that GAP-LSTM trades a linear increase in computational cost over a simpler neural network approach such as SVD-LSTM and Attention-LSTM, and a slight increase over state-of-the-art approaches such as GWN, MTGNN, Triformer, and ESG in exchange for higher forecasting performance. The RGSL method presents a higher execution time due to its layer complexity. Autoregressive models like ARIMA are unable to exploit information from all nodes simultaneously, resulting in a suboptimal forecasting performance and a considerably higher computational cost.

As an additional analysis, we report the GAP-LSTM space and time complexity with an increasing number of nodes in the sensor network. Results in Table VII show the amount of memory and time (absolute, relative, and per-node) required for model training, considering the Wind NREL dataset (originally including five nodes) and a progressively increasing number of synthetically added nodes in the network. We observe that the absolute time (expressed in seconds per epoch) increases sublinearly with the number of nodes. This result is also more intuitively represented by relative times. For instance, increasing the sensor network by a factor of 10 (from 5 to 50 nodes) results in a time increase of 6.83×. As for memory consumption, results for absolute memory (expressed in MBs) show a negligible increase as the number of nodes grows. In relative terms, increasing the sensor network size by a factor of 2 (from 5 to 10 nodes) only results in a 1% memory increase. In the largest case, where the network size is increased by factor of 10 (from 5 to 50 nodes) the memory increase observed is just 31%. This result is expected, since the core components and the number of model parameters remain constant with the addition of new nodes, with the exception of the adjacency matrix and attention heads, which are extended, leading to a logarithmic increase in space. These results show that GAP-LSTM makes an efficient use of computational resources, resulting in a minimal overhead with the increase of network size, which is a typical scenario in

TABLE VIII
NUMBER OF TRAINABLE PARAMETERS FOR EACH MODEL, MEASURED ON LIGHTSOURCE WITH A BATCH SIZE OF 16 FOR ALL MODELS. THE NUMBER OF PARAMETERS PRESENTS MINIMAL VARIATIONS ACROSS DATASETS

Method	Number of Parameters
LSTM	1,954
GRU	1,734
Bi-LSTM	1,954
Attention-LSTM	2,086
CNN-LSTM	1,954
SVD-LSTM	1,954
GCN-LSTM	2,086
GraphWaveNet	292,895
MTGNN	132,787
Triformer	161,403
ESG	155,529
RGSL	754,811
GAP-LSTM-Default	39,993
GAP-LSTM-Weighted	43,689
GAP-LSTM-GAT	47,385

real-world sensor networks. Additional information pertaining to model complexity is shown by the number of trainable parameters, as shown in Table VIII. Results highlight that GAP-LSTM is competitive in terms of model complexity (in terms of the number of parameters) when compared to its direct competitors.

V. CONCLUSION

This article proposes GAP-LSTM, a novel method for geo-distributed forecasting that focuses on exploiting spatio-temporal autocorrelation in multivariate data generated by multiple nodes. Existing approaches for this challenging task either do not simultaneously take into account the spatial and the temporal dimensions of data, or do not preserve the learned spatio-temporal patterns throughout the entire downstream forecasting task. Our method leverages the synergic interaction of graph convolution, attention-based LSTM, 2-D convolution, and latent memory states to overcome these limitations.

An extensive evaluation involving real-world datasets on traffic, energy, and pollution domains shows that GAP-LSTM outperforms state-of-the-art methods. An ablation study shows that all components bring a positive contribution to this outcome. The method also provides a visualization that allows domain experts to gather additional insights about predictions and supports them in their decisions. In future work, we aim to explore the adaptability of our method in applications with multiple correlated nodes and without an explicit geo-distributed network. Moreover, we will investigate the explicit treatment of specific forms of spatio-temporal autocorrelation in the loss function of the model. Finally, we aim to assess our model's accuracy in other domains where very short-term and long-term forecasting horizons are required.

REFERENCES

[1] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis, Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.

[2] S. Makridakis and M. Hibon, "ARMA models and the box–Jenkins methodology," *J. Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[3] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[4] C. Bergmeir, I. Triguero, D. Molina, J. L. Aznarte, and J. M. Benitez, "Time series modeling and forecasting using memetic algorithms for regime-switching models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 11, pp. 1841–1847, Nov. 2012.

[5] M. Favereau, Á. Lorca, M. Negrete-Pincetic, and S. Vicuña, "Robust streamflow forecasting: A student's T-mixture vector autoregressive model," *Stochastic Environ. Res. Risk Assessment*, vol. 36, no. 11, pp. 3979–3995, Nov. 2022.

[6] M. Eren, "Fuzzy autoregressive distributed lag model-based forecasting," *Fuzzy Sets Syst.*, vol. 459, pp. 82–94, May 2023.

[7] E. Lazar and X. Xue, "Forecasting risk measures using intraday data in a generalized autoregressive score framework," *Int. J. Forecasting*, vol. 36, no. 3, pp. 1057–1072, Jul. 2020.

[8] Y. Zhao, L. Ye, P. Pinson, Y. Tang, and P. Lu, "Correlation-constrained and sparsity-controlled vector autoregressive model for spatio-temporal wind power forecasting," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5029–5040, Sep. 2018.

[9] J. W. Messner and P. Pinson, "Online adaptive lasso estimation in vector autoregressive models for high dimensional wind power forecasting," *Int. J. Forecasting*, vol. 35, no. 4, pp. 1485–1498, Oct. 2019.

[10] X. Liu, Z. Lin, and Z. Feng, "Short-term offshore wind speed forecast by seasonal ARIMA—A comparison against GRU and LSTM," *Energy*, vol. 227, Jul. 2021, Art. no. 120492.

[11] M. Yurtsever, "Unemployment rate forecasting: LSTM-GRU hybrid approach," *J. Labour Market Res.*, vol. 57, no. 1, pp. 1–9, Jun. 2023.

[12] K. Bandara, C. Bergmeir, and H. Hewamalage, "LSTM-MSNet: Leveraging forecasts on sets of related time series with multiple seasonal patterns," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1586–1599, Apr. 2021.

[13] G. Dudek, P. Pelka, and S. Smyl, "A hybrid residual dilated LSTM and exponential smoothing model for midterm electric load forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2879–2891, Jul. 2022.

[14] A. Glowacz, "Thermographic fault diagnosis of electrical faults of commutator and induction motors," *Eng. Appl. Artif. Intell.*, vol. 121, May 2023, Art. no. 105962.

[15] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proc. NIPS*, vol. 34, Dec. 2021, pp. 22419–22430.

[16] K. Zhu, Y. Li, W. Mao, F. Li, and J. Yan, "LSTM enhanced by dual-attention-based encoder–decoder for daily peak load forecasting," *Electric Power Syst. Res.*, vol. 208, Jul. 2022, Art. no. 107860.

[17] S. Du, T. Li, Y. Yang, and S. J. Horng, "Multivariate time series forecasting via attention-based encoder–decoder framework," *Neurocomputing*, vol. 388, pp. 269–279, May 2020.

[18] M. Ceci, R. Corizzo, D. Malerba, and A. Rashkovska, "Spatial autocorrelation and entropy for renewable energy forecasting," *Data Mining Knowl. Discovery*, vol. 33, no. 3, pp. 698–729, May 2019.

[19] M. Ceci, R. Corizzo, F. Fumarola, D. Malerba, and A. Rashkovska, "Predictive modeling of PV energy production: How to set up the learning task for a better prediction?" *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 956–966, Jun. 2017.

[20] R. Corizzo, M. Ceci, H. Fanaee-T, and J. Gama, "Multi-aspect renewable energy forecasting," *Inf. Sci.*, vol. 546, pp. 701–722, Feb. 2021.

[21] J. Li, S. Wei, and W. Dai, "Combination of manifold learning and deep learning algorithms for mid-term electrical load forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2584–2593, May 2023.

[22] S. Du, T. Li, Y. Yang, and S.-J. Horng, "Deep air quality forecasting using hybrid deep learning framework," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2412–2424, Jun. 2021.

[23] J. Qu, Z. Qian, and Y. Pei, "Day-ahead hourly photovoltaic power forecasting using attention-based CNN-LSTM neural network embedded with multiple relevant and target variables prediction pattern," *Energy*, vol. 232, Oct. 2021, Art. no. 120996.

[24] Q. Pan, W. Hu, and N. Chen, "Two birds with one stone: Series saliency for accurate and interpretable multivariate time series forecasting," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 2884–2891.

[25] R. Yan, J. Liao, J. Yang, W. Sun, M. Nong, and F. Li, "Multi-hour and multi-site air quality index forecasting in Beijing using CNN, LSTM, CNN-LSTM, and spatiotemporal clustering," *Exp. Syst. Appl.*, vol. 169, May 2021, Art. no. 114513.

[26] C. Ren, L. Jia, and Z. Wang, "A CNN-LSTM hybrid model based short-term power load forecasting," in *Proc. Power Syst. Green Energy Conf. (PSGEC)*, Aug. 2021, pp. 182–186.
[27] X. Shao, C. Pu, Y. Zhang, and C. S. Kim, "Domain fusion CNN-LSTM for short-term power consumption forecasting," *IEEE Access*, vol. 8, pp. 188352–188362, 2020.
[28] Y. Ding, Y. Zhu, J. Feng, P. Zhang, and Z. Cheng, "Interpretable spatio-temporal attention LSTM model for flood forecasting," *Neurocomputing*, vol. 403, pp. 348–359, Aug. 2020.
[29] Z. Li, Q. Ren, L. Chen, J. Li, and X. Li, "Multi-scale convolutional networks for traffic forecasting with spatial–temporal attention," *Pattern Recognit. Lett.*, vol. 164, pp. 53–59, Dec. 2022.
[30] X. He, S. Shi, X. Geng, L. Xu, and X. Zhang, "Spatial–temporal attention network for multistep-ahead forecasting of chlorophyll," *Appl. Intell.*, vol. 51, pp. 4381–4393, 2021.
[31] T. Misiakiewicz and S. Mei, "Learning with convolution and pooling operations in kernel methods," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 29014–29025.
[32] C. Sun, Y. Ning, D. Shen, and T. Nie, "Graph neural network-based short-term load forecasting with temporal convolution," *Data Sci. Eng.*, pp. 1–20, Nov. 2023, doi: 10.1007/s41019-023-00233-8.
[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[34] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.
[35] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell., (IJCAI)*, 2018, pp. 3634–3640.
[36] W. Zhong, Q. Suo, X. Jia, A. Zhang, and L. Su, "Heterogeneous spatio-temporal graph convolution network for traffic forecasting with missing values," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 707–717.
[37] M. Li, S. Chen, Y. Shen, G. Liu, I. W. Tsang, and Y. Zhang, "Online multi-agent forecasting with interpretable collaborative graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4768–4782, Apr. 2024.
[38] Z. Wu, D. Zheng, S. Pan, Q. Gan, G. Long, and G. Karypis, "TraverseNet: Unifying space and time in message passing for traffic forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 2003–2013, Feb. 2024.
[39] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2017, vol. 31, no. 1, pp. 1–12.
[40] C. Luo, X. Li, and Y. Ye, "PFST-LSTM: A SpatioTemporal LSTM model with pseudoflow prediction for precipitation nowcasting," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 843–857, 2021.
[41] X. Ren and S. Yuan, "GCN-LSTM combined model for urban link mean speed prediction in the regional traffic network," in *Proc. IEEE Intl Conf Dependable, Autonomic Secure Comput., Intl Conf Pervasive Intell. Comput., Intl Conf Cloud Big Data Comput., Intl Conf Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Sep. 2022, pp. 1–7.
[42] Y. Shi, Y. Wang, Y. Qu, and Z. Chen, "Integrated GCN-LSTM stock prices movement prediction based on knowledge-incorporated graphs construction," *Int. J. Mach. Learn. Cybern.*, vol. 15, no. 1, pp. 161–176, Jan. 2024.
[43] X. Li et al., "A shortcut enhanced LSTM-GCN network for multi-sensor based human motion tracking," *IEEE Trans. Autom. Sci. Eng.*, early access, pp. 1–10, 2004, doi: 10.1109/TASE.2023.3307890.
[44] R.-G. Cirstea, C. Guo, B. Yang, T. Kieu, X. Dong, and S. Pan, "Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting," in *Proc. Thirty-First Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 1994–2001.
[45] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial–temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.
[46] H. Yu et al., "Regularized graph structure learning with semantic knowledge for multi-variates time-series forecasting," in *Proc. Thirty-First Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 2362–2368.
[47] W. Shao et al., "Long-term spatio-temporal forecasting via dynamic multiple-graph attention," in *Proc. Thirty-First Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 2225–2232.
[48] D. Wu and W. Lin, "Efficient residential electric load forecasting via transfer learning and graph neural networks," *IEEE Trans. Smart Grid*, vol. 14, no. 3, pp. 2423–2431, May 2023.
[49] M. Khodayar, G. Liu, J. Wang, O. Kaynak, and M. E. Khodayar, "Spatiotemporal Behind-the-Meter load and PV power forecasting via deep graph dictionary learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4713–4727, Oct. 2021.
[50] J. Ye et al., "Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 2296–2306.
[51] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 753–763.
[52] H. Li, C. A. Calder, and N. Cressie, "Beyond Moran's I: Testing for spatial dependence based on the spatial autoregressive model," *Geographical Anal.*, vol. 39, no. 4, pp. 357–375, Oct. 2007.
[53] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.
[54] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.
[55] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in Beijing," *Proc. Roy. Soc. A: Math., Phys. Eng. Sci.*, vol. 473, no. 2205, Sep. 2017, Art. no. 20170457.

Massimiliano Altieri is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Bari, Bari, Italy.

His research interests and activity mainly focuses on the design of novel deep learning methods for time series forecasting involving graph convolutional neural networks.

Mr. Altieri also served as a reviewer for a wide range of known conferences in the field of machine learning and data mining.

Roberto Corizzo (Member, IEEE) received the Ph.D. degree in computer science from the University of Bari Aldo Moro, Bari, Italy, in 2018.

He was a Research Fellow with the Department of Computer Science, University of Bari, Bari. He is currently an Assistant Professor with the Department of Computer Science, American University, Washington, DC, USA. He has coauthored 50 articles, including 13 publications in journals, such as IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, *Neural Networks*, and *Machine Learning*.

Dr. Corizzo participated in the scientific committee of international conferences and served as a reviewer for several international journals.

Michelangelo Ceci received the Ph.D. degree in computer science from the University of Bari Aldo Moro, Bari, Italy, in 2005.

He is currently a Full Professor of computer science with the University of Bari, Bari. He has authored more than 170 papers in journals and conferences on machine learning and data mining. He has been the unit coordinator of EU and national projects.

Dr. Ceci has been in the PC of many conferences, e.g., IEEE International Conference on Data Mining (ICDM), SIAM International Conference on Data Mining (SDM), International Joint Conference on Artificial Intelligence (IJCAI), and Association for the Advancement of Artificial Intelligence (AAAI). He was the PC Co-Chair for SEBD2007, Discovery Science 2016, ISMIS 2018, and ISMIS 2022 and the General Chair for ECML-PKDD 2017. He is an Associate Editor of Data Mining and Knowledge Discovery (DMKD) and *Machine Learning Journal* (MLJ), and an Editorial Board (EB) Member of *Journal of Intelligent Information Systems* (JIIS).

Open Access funding provided by 'Università degli Studi di Bari "Aldo Moro"' within the CRUI CARE Agreement