Applied Intelligence (2024) 54:3918–3932
https://doi.org/10.1007/s10489-023-05220-0

Learning evolving relations for multivariate time series forecasting

Binh Nguyen-Thai[1,2] · Vuong Le[3] · Ngoc-Dung T. Tieu[4,5] · Truyen Tran[6] · Svetha Venkatesh[6] · Naeem Ramzan[1]

Accepted: 6 December 2023 / Published online: 15 March 2024
© The Author(s) 2024

Abstract
Multivariate time series forecasting is essential in various fields, including healthcare and traffic management, but it is a challenging task due to the strong dynamics in both *intra-channel relations* (temporal patterns within individual variables) and *inter-channel relations* (the relationships between variables), which can evolve over time with abrupt changes. This paper proposes ERAN (Evolving Relational Attention Network), a framework for multivariate time series forecasting, that is capable to capture such dynamics of these relations. On the one hand, ERAN represents inter-channel relations with a graph which evolves over time, modeled using a recurrent neural network. On the other hand, ERAN represents the intra-channel relations using a temporal attentional convolution, which captures the local temporal dependencies adaptively with the input data. The elvoving graph structure and the temporal attentional convolution are intergrated in a unified model to capture both types of relations. The model is experimented on a large number of real-life datasets including traffic flows, energy consumption, and COVID-19 transmission data. The experimental results show a significant improvement over the state-of-the-art methods in multivariate time series forecasting particularly for non-stationary data.

Keywords Time series forecasting · Multivariate time series forecasting · Dynamic graph neural networks · Attention mechanism

1 Introduction

In this paper, we study the problem of multivariate time series forecasting, which is to predict the future data points given previous data of time series. Forecasting time series is critical for many real-life applications, including predicting traffic flow, electricity consumption, and COVID transmission. Accurate forecasting is crucial for making informed decisions and planning for the future.

Multivariate time series forecasting is challenging due to the complexity in both intra- and inter-channel relations. Intra-channel relations involve the temporal patterns within individual variables, determining the dependencies of future values on prior ones. Intra-channel relations can be highly dynamic, particular in non-stationary time series, making it difficult to forecast. For example, in COVID-19 data, changes in government policies may impact COVID-19 transmission in a city, resulting in non-stationarity in temporal patterns and making it challenging to predict new cases. Inter-channel relations, on the other hand, refer to the dependencies between pairs of variables. Again, these relations can evolve over time. For instance, the correlation between the spread of COVID-19 among cities or countries may vary over time due to adaptive government policies such as social distancing or border closures. Therefore, accurately capturing the dynamics of these types of relations is essential for multivariate time series forecasting.

Time series forecasting have been extensively studied in conventional models such as autoregressive model (AR), moving average model (MA), autoregressive integrated moving average (ARIMA) [15]. Recently, deep neural network-based models, such as recurrent neural networks (RNNs) [7, 14], and convolutional neural networks (CNNs) [3], have shown promising results due to their ability to capture nonlinear temporal patterns. These models represent the dependencies of future data points on previous data points using a set of learnable parameters. However, since these paramerters are fixed after training, they can only capture

The source code is available at the following URL: https://zenodo.org/records/10528296.

✉ Binh Nguyen-Thai
thaibinh.nguyen@uws.ac.uk; nguyenthaibinh@gmail.com

Extended author information available on the last page of the article

invariant temporal patterns, and therefore, are insufficient to model time series with time-varying patterns, such as non-stationary time series that are commonly observed in reality.

To model the inter-channel relations, recent works have been applying graph neural networks to multivariate time series data. In this approach, a multivariate time series is viewed as a graph, with each variable represented as a node and the underlying correlations between variables represented as connections between nodes. The graph can be either pre-defined or learned from data. By combining a graph neural network with a temporal model such as an RNN [2] or a CNN [12, 34, 35], both types of relations can be modeled simultaneously. The primary limitation of the this approach is its static graph structure, where a single graph is used throughout the entire lifespan of the time series. However, the static graph structure is inadequate for capturing the evolving relationships between variables over time, as the example of COVID transmission above.

A highly plausible path to tackle these challenges is via *learning the evolution of the spatio-temporal relations in time series.* Unlike entities in static data, time series variables have unique evolving lives throughout space-time. As a series progresses, it changes its internal states and interacts with other series at arbitrary time. Guided by these principles, we introduce ERAN (Evolving Relational Attention Network), a novel method for modeling multivariate time series. ERAN learns to extract the graph structure underlying inter-channel relations at each time step. In ERAN, a graph structure is represented by an adjacency matrix. The evolution of the adjacency matrices is modelled by a recurrent neural network, where the adjacency matrix at each time step depends on that of the previous step and the current observed data, allowing ERAN to accurately model the evolving inter-channel relations in the data.

Once discovered, the dynamic graph structure guides a reasoning process that jointly captures the intra- and inter-channel relations of a time series. This process happens within a multi-layer architectures that alternate between inter-channel relations by graph convolution network and intra-channel relations by a fused Temporal Attentional Convolution (TAC). This process results in a concrete representation of the observed time series which facilitates convenient decoding into the future forecast.

The ERAN model stands out with its authentic and explicit modeling of evolving relations in time series leading to the effective and stable forecasting process. These advantages are demonstrated through a comprehensive set of experiments across multiple domains, including traffic flow, electricity consumption, and COVID-19 transmission. ERAN consistently outperforms existing state-of-the-art models on these tasks. Analysis of the model reveals its effectiveness in exploring the underlying dynamic relations during the forecasting process.

In summary, we present the following innovations:

- A method to model the dynamics of inter-channel relations in multivariate time series, by learning a dynamic graph underlying such relations, that evolves over time.
- Temporal attentional convolution, a self-attention mechanism operating on sliding windows, to model the dynamics of intra-channel relations in multivariate time series.
- Development of a framework for multivariate time series forecasting that concurrently captures the dynamics of intra- and inter-channel relations between variables.
- Conducting extensive experiments on multiple real-world datasets to demonstrate the effectiveness of explicitly considering the dynamics of both relations in multivariate time series forecasting, in particular, for non-stationary time series.

2 Related work

2.1 Traditional time series forecasting

Time series forecasting has been studied for decades. Traditional time series techniques are mainly based on the statistical approach. These methods include the well-known autoregressive integrated moving average (ARIMA) [15], support vector regression (SVR) [25], random forest (RF) [10], vector autoregressive models (VARs) [15].

ARIMA is a generalisation of the autoregressive (AR) and and autoregresive moving average (ARMA) models [22]. SVR [15] is a type of support vector machine that can perform regression by finding a hyperplane that fits the data with the maximum margin. SVR can handle non-linear and high-dimensional data, and provide probabilistic forecasts. RF [10] is an ensemble learning method that can perform regression by combining multiple decision trees that are trained on different subsets of the data. RF can handle noisy and heterogeneous data, and reduce the variance and overfitting of individual trees.

Vector autoregresive models (VARs) extend AR and ARMA to extract linear correlation between variables in multivariate time series [15]. Nevertheless, traditional methods have several drawbacks. Firstly, they are linear models and hence cannot capture non-linear dependencies present in complex data. Secondly, these methods train time series individually, and therefore, they are not scalable to large-scale data sets containing millions of time series. Furthermore, they cannot leverage the common patterns shared between time series in the data set, due to their individual training.

2.2 Deep learning-based time series forecasting

Deep learning-based methods have recently shown promising results in time series forecasting by capturing non-linear dependencies in the data. Among these techniques, RNN and its variants LSTM and GRU [6, 7, 14], has been used, such as Deep AR [30], Deep State-Space Models (DSSM) [28], TimeGrad [29] to name a few. CNN-based models such as WaveNet [24], GluonTS [1] have also demonstrated their effectiveness in modeling time series data.

Recently, transformers (e.g., [32]), renowned for their adeptness in modeling long-range dependencies and interactions within sequential data, have been increasingly utilised in time series forecasting [17, 19, 21, 33, 38]. LogSparse Transformer [19] proposes the LogSparse attention that to reduce the computation complexity of the original Transformer. Reformer [17]. Informer [38] proposes the sparsity of attention score through KL divergence estimation and proposes ProbSparse self-attention which achieves O(Llog L) complexity. Crossformer [37]. Temporal Fusion Transformer [21].

These models embed the observed multivariate time series into a sequence of vectors in a shared hidden space, and use RNN, CNN, or self-attention mechanisms to model the sequence in the temporal axis. Since the variables are encoded in a shared hidden space, the inter-dependencies between variables are not modeled explicitly.

2.3 Graph neural network for time series forecasting

Graph neural networks (GNN) have been showing great successes in structured data like social network, protein networks, chemical networks, and human skeleton data. The main goal of a GNN is to capture the dependencies of nodes via a graph structure. GNN can learn the representation of a node by leveraging not only that node's feature but also its neighbor nodes. Various techniques have been proposed for this purpose such as graph convolution [4, 8, 16] and message passing [11, 23, 27].

Inspired by the success of the GNN in other domains, researchers have recently applied GNN to multivariate time series such as Graph Wavenet (GWNet) [34], Spatio-Temporal Graph Convolutional Networks (STGCN) [36] Attention-based Spatial-temporal Graph Convolutional Network (ASTGCN) [12]. A multivariate time series can be seen as a graph where variables correspond to nodes of the graph, and the edges of the graph are the underlying dependencies between the variables. By combining GNN and a temporal model, e.g., a RNN or a CNN, these works can learn time series representations that capture both intra- and inter-channel relations. However, these models need a pre-defined graph, which is not always readily available.

To enable the GNN on data where a pre-defined graph is unavailable, researchers have proposed to learn latent structure from data. In time series, a number of models have been proposed to learn an underlying graph from data such as MTGNN [35], Adaptive Graph Convolutional Recurrent Network (AGCRN) [2], Spectral Temporal Graph Neural Network (StemGNN) [5]. The main disadvantage is that they use a static graph structure over the entire time span of a time series, thus cannot capture the dynamic dependencies between variables.

3 Proposed method

3.1 Problem formulation

A multivariate time series is represented by a matrix $Y = [X_1, X_2, \ldots, X_T] \in \mathbb{R}^{N \times T}$, where T is the number of time steps and N is the number of variables. In this notation, $X_t \in \mathbb{R}^N$ represents a slice of Y observed at time step t. Given a historical window of L observed time steps, $Y^o = \left[X_{T-L+1}, X_{T-L+2}, \ldots, X_T\right]$, and a forecast horizon τ, the task is to predict the values of the next τ steps in the future: $Y^f = \left[X_{T+1}, X_{T+2}, \ldots, X_{T+\tau}\right]$. Often, historical data may be associated with covariates such as date, time, and location. Therefore, we assume that the input data for forecasting Y^f is $\mathcal{S} = \left[S_{T-L+1}, S_{T-L+2}, \ldots, S_T\right]$, where $S_t \in \mathbb{R}^{D_{in} \times N}$ is obtained by concatenating X_t and its covariates, D_{in} is the dimensionality of the input features. Our goal is to build a model $\mathcal{F}$ that predicts Y^f from $\mathcal{S}$.

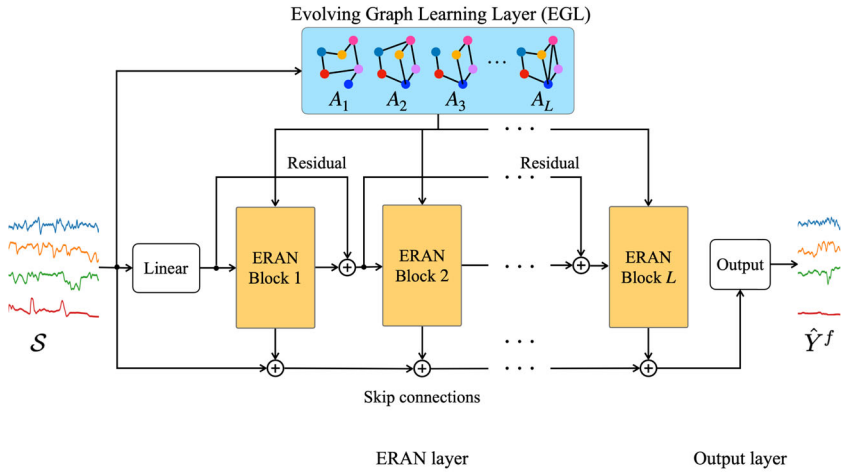
$$\hat{Y}^f = \mathcal{F}(\mathcal{S}, \Theta) \tag{1}$$

Here, $\hat{Y}^f \in \mathbb{R}^{N \times \tau}$ is the predicted values in the forecasting horizon, and Θ is the set of all model parameters.

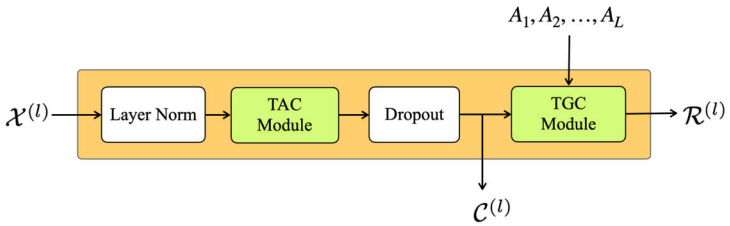
3.2 Model overview

We propose a novel approach to address the multivariate time series forecasting problem by leveraging a dynamic graph $\mathcal{G}_t = (\mathcal{V}, A_t)$ that captures the interactions between the variables at each time step of the series. The set of nodes $\mathcal{V}$ is of size $\mathcal{V} = N$, and $A_t \in \mathbb{R}^{N \times N}$ is the adjacency matrix whose entries reflect the strengths of the relations between pairs of variables at the t-th time step. It is worth noting that this graph is not pre-defined. Instead, our proposed model will learn to extract the node features and generate the corresponding adjacency matrix A_t and its evolution over time steps.

The proposed approach is implemented in ERAN, whose overall architecture is illustrated in Fig. 1a. Firstly, the multivariate time series is input into a *Evolving Graph Learning layer* (EGL) to generate a sequence of adjacency matrices,



(a) High-level architecture of ERAN



(b) Architecture of an ERAN Block

Fig. 1 (a) The high level architecture of ERAN model. The ERAN is composed of an Evolving Graph Learning layer (EGL) which learns to generate the evolving adjacency matrices, and multiple ERAN blocks stacked together. Residual connections and skips connections are used to prevent vanishing gradient. (b) An ERAN block consists of two main components: a Temporal Attentional Convolution (TAC) and a Temporal Graph Convolution (TGC) which are integrated to capture the intra- and inter-channel relations

one for each time step. This sequence of matrices serves as the input for the *ERAN layer*, which is composed of multiple *ERAN blocks* stacked together. An ERAN block is presented in Fig. 1b. Each ERAN block comprises two main components: a *Temporal Attentional Convolution* (TAC) and a *Temporal Graph Convolution* (TGC), which jointly capture both intra- and inter-channel relations. Furthermore, each ERAN block includes a LayerNorm and Dropout layer to prevent overfitting. To prevent vanishing gradients and accelerate training, residual and skip connections are utilised throughout all ERAN blocks.

Each ERAN block has two output branches: (i) the *residual output* $\mathcal{R}^{(i)}$, which serves as part of the input for the next ERAN block, and (ii) the *skip output* $\mathcal{C}^{(i)}$, which is fed directly to the output layer (described in the next paragraph) to contribute to the forecast.

The *output layer* utilises the skip connection outputs of the ERAN blocks to generate future values. The model is trained to produce the most accurate prediction of these future values. The Mean Absolute Error is chosen as the objective function to train ERAN, which is computed as follows:

$$\mathcal{L}(\hat{Y}^f, Y^f, \Theta) = \frac{1}{N.\tau} \sum_{t=1}^{T} |\hat{Y}^f - Y^f| \tag{2}$$

where Y^f is the ground truth of the forecasting horizon, $\hat{Y}^f$ are the values predicted by the model, and Θ represents all model parameters.

The detailed design of these components is described in the subsequent sections.

3.3 EGL: Evolving Graph Learning layer

In a GNN-based method, the adjacency matrix plays a central role in learning the relationships between individual variables. In deterministic systems, this matrix can be predefined based on human knowledge. For instance, in traffic flow data, the adjacency matrix can be constructed from the road network and the distances between the sensors. However, in many cases, such a graph is not readily available or too complex to be defined manually. Early learning-based works proposed generating such a structure from the data without prior knowledge of the graph [2, 5, 35]. However, these works generate a single adjacency matrix to cover the relations between variables throughout the entire time series, and thus they are not adaptive to the dynamics of the data. In contrast, EGL learns a series of adjacency matrices, each for a time step, enabling the model to capture the evolving relationships of individual variables.

Now we present how EGL works. Our aim is to design a recurrent process that generates A_t, the adjacency matrix

at time step t, conditioned on the previous step's adjacency matrix A_{t-1} and current input value X_t. However, directly modeling the values of the adjacency matrices would require $\mathcal{O}\left(N^4\right)$ computational complexity and memory consumption[1], which is remarkably expensive for a large network. To reduce the computational complexity, we factorise the matrix into two low-rank matrices $H_t, H't \in \mathbb{R}^{de \times N}$, where $d_e \ll N$, holding the states of N variables at time step t. Instead of generating A_t, EGL learns to generate H_t and H'_t and approximate A_t by $A_t = g\left(H_t, H'_t\right)$. In our implementation, we choose the function $A_t = relu\left(tanh\left(H_t^\top H'_t\right)\right)$. By using the $relu$ and $tanh$ function, we aim to make the matrix A_t sparse, forcing the model to retain only important relations. The benefits of this factorisation are not limited to reducing computational complexity but also to maintaining the low-rank properties of A_t.

We use a GRU (Gated Recurrent Unit), a type of recurrent neural network, to model the evolution of H_t, where H_t is the hidden state of the GRU, and the layer input is the GRU's input. Similarly, we use another GRU to model the evolution of H'_t. Specifically, first, we embed X_t into embedding vectors E_t and E'_t using linear transformations $E_t = W_E X_t \in \mathbb{R}^{d_e \times N}$ and $E'_t = W'_E X_t \in \mathbb{R}^{d_e \times N}$, where $E_t, E'_t \in \mathbb{R}^{d_e \times N}$ are learnable parameters. Then, E_t and E'_t are fed into the GRU to compute hidden states H_t and H'_t as follows:

$$H_t = GRU\left(E_t, H_{t-1}\right), \tag{3}$$

$$H'_t = GRU\left(E't, H'_{t-1}\right), \tag{4}$$

Since a GRU originally operates on vectors, the GRU's design need to be updated to operate on matrices X_t, H_t, and H'_t. For this purpose, we use the same approach presented in [26]. In detail, the GRU's computation is presented in Algorithm 1. Here, $WZ, W_R, U_Z, U_R, W_H, U_H, B_Z, B_R, B_H \in \mathbb{R}^{d_e \times d_e}$ are learnable parameters.

Algorithm 1 Matrix GRU.

function: $H_t = GRU\left(X_t, H_{t-1}\right)$
input: $E_t \in \mathbb{R}^{d_e \times N}, H_{t-1} \in \mathbb{R}^{d_e \times N}$
output: $H_t \in \mathbb{R}^{d_e \times N}$
begin:
$Z_t = \text{sigmoid}\left(W_Z E_t + U_Z H_{t-1} + B_Z\right)$
$R_t = \text{sigmoid}\left(W_R E_t + U_R H_{t-1} + B_R\right)$
$\tilde{E}_t = \tanh\left(W_H E_t + U_H (R_t \circ H_{t-1}) + B_H\right)$
$H_t = (1 - Z_t) \circ H_{t-1} + Z_t \circ \tilde{H}_t$
return H_t

The dynamic adjacency matrices built from this process contain the evolving relational structure between variables and will be used to extract the intra- and inter-channel relations, which will be described in Section 3.4.

[1] Flattening the matrix results in a vector of length N^2, and mapping between two consecutive matrices requires N^4 operations.

3.4 ERAN block

The ERAN layer layer consists of multiple ERAN blocks which are stacked together to form a multi-layer network with skip connection. Each ERAN block is designed to capture the intra- and inter-channel relations by employing a Temporal Attentional Convolution (TAC) and a Temporal Graph Convolution (TGC). In this section, we will introduce the motivation behind an ERAN block and its architecture.

3.4.1 TAC: Temporal Attentional Convolution Module

For local temporal pattern modeling, convolutional neural networks (CNNs) are a common choice to find local motifs. CNNs learn a kernel to operate on a sliding window and compute the output from input within a context. In CNN, one unique kernel is applied for the whole lifetime of a time series. However, a unique kernel is insufficient to capture the temporal dynamics where different parts of the same time series may have changing temporal patterns. Furthermore, once learned, the kernel is fixed, thus it is poor at capturing the temporal patterns when the test set and training set have different patterns.

In order to model such temporal dynamics, here we design temporal attentional convolution (TAC). Unlike CNN, which aggregates input within a local context to compute the output using a fixed kernel, TAC employs a self-attention mechanism to learn to focus on important input when computing the output, enabling it to capture changes in temporal patterns over time. Attention mechanisms with the ability to learn to focus on important parts within a context have shown their effectiveness in natural language processing and computer vision. We extend that concept to multivariate time series where we need to deal with N sequences corresponding to N variables.

For each sequence, the data point at each time step will attend to its neighbors of the same sequence, within a given window with of size w ($w = 3, 5, 7, \dots$) where each time step attends w time steps each size and itself. Specifically, at time step t, the attention region is $\mathcal{N}_w(t) = \left\{t', t - \frac{w-1}{2} \le t' \le t + \frac{w-1}{2}\right\}$. This approach is in contrast to global attention, where each data point attends to all data points across all time steps. By limiting attention to a local region, the proposed mechanism reduces the computational complexity and memory consumption required for processing the sequence data. An example of TAC operating in a window of size $w = 5$ is illustrated in Fig. 2.

Given the input $\mathbf{x}_{i,t}$ of the i-th series at time step t, a single-headed attention for the output feature $\mathbf{z}_{i,t} \in \mathbb{R}^{d_{out}}$ is

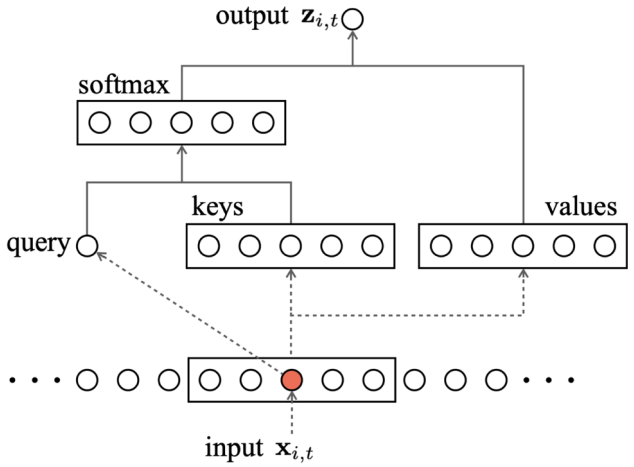


Fig. 2 An example of the TAC module over a window of size $w = 5$

computed as:

$$\mathbf{z}_{i,t} = \sum_{t' \in \mathcal{N}_w(t)} \left(\mathbf{q}_{i,t}^\top \mathbf{k}_{i,t'} + \mathbf{q}_{i,t}^\top \boldsymbol{\alpha}_{t'-t} \right) \mathbf{v}_{i,t'} \tag{5}$$

where the queries $\mathbf{q}_{i,t}$, keys $\mathbf{k}_{i,t'}$, and values $\mathbf{v}_{i,t'}$ are linear transformations of $\mathbf{x}_{i,t}$ and its neighbourhoods, and are computed as follows.

$$\mathbf{q}_{i,t} = W_Q \mathbf{x}_{i,t}, \mathbf{k}_{i,t'} = W_K \mathbf{x}_{i,t'}, \mathbf{v}_{i,t'} = W_V \mathbf{x}_{i,t'} \tag{6}$$

Here, $W_Q, W_K, W_V \in \mathbb{R}^{d_{out} \times d_{in}}$ are learnable parameters, and $\boldsymbol{\alpha}_{t'-t} \in \mathbb{R}^{d_{out}}$ represents the relational position embedding associated with the neighborhood t' of t. The relational position embedding is introduced here to capture the temporal information of the input. It was introduced in [31], where its effectiveness over absolute position embeddings was suggested.

The way in which we compute $\mathbf{z}_{i,t}$ (see (5)) is similar to that of a convolutional operator across the temporal dimension. However, instead of using a fixed kernel, the kernel weights are computed from the content of the variables and their underlying dependencies, making the model adaptive to the dynamism of the temporal patterns of the data.

Multi-headed TAC In practice, multiple attention heads can be used to calculate different representation sets from the input. To achieve this, each single-headed attention uses linear transforms $W_Q^{(h)}, W_K^{(h)}, W_V^{(h)} \in \mathbb{R}^{d_{in} \times d_{out}/H}$ to generate the output $\mathbf{z}_{i,t}^{(h)} \in \mathbb{R}^{d_{out}/H}$, where H is the number of attention heads. These outputs are then concatenated to obtain the final output $\mathbf{z}_{i,t}$ of the multi-headed attention: $\mathbf{z}_{i,t} = \left[\mathbf{z}_{i,t}^{(1)}; \mathbf{z}_{i,t}^{(2)}; \ldots; \mathbf{z}_{i,t}^{(H)} \right] \in \mathbb{R}^{d_{out}}$.

Dilated TAC When m ERAN blocks are stacked together, the receptive field will be $m\,(w-1)$. To further increase the receptive field, we use a "dilated" window, where neighbor data points can be skipped regularly. This concept is similar to the dilated convolution presented in [24]. We use a dilation factor p to control the dilation of each layer, where the dilation of the i-th ERAN block is p^{i-1}. For example, if the dilation factor is 2, then the dilation of the first layer is 1, the second layer is 2, the third layer is 4, and so on. In summary, in a network of m ERAN blocks, the receptive field is:

$$RF = \begin{cases} m(w-1) & \text{if } p = 1 \\ \left[1 + (w-1)(p^l - 1)\right]/(p-1) & \text{if } p > 1 \end{cases} \tag{7}$$

assuming the window size w is fixed for all layers.

3.4.2 TGC: Temporal graph convolution module

Graph convolutional networks (GCN) generalise convolutional neural networks (CNNs) to work on graph-structured data, such as social networks or protein structures [4, 8, 16]. GCNs generate output node features that capture the spatial dependencies of nodes, given their node features and a graph structure. While there are existing works that use graph convolutions for time series forecasting, most of them use a pre-defined graph or a static graph learned from data for the entire time series. In contrast, we use an evolving graph for the time series, where each time step has a different adjacency matrix computed in the EGL layer. At each step, graph convolutions are performed with the corresponding adjacency matrix.

There are many ways to perform graph convolutions, such as spectral graph convolution [4], graph convolutional networks [16], and approximation of convolutions using Chebyshev polynomials [8]. Here, we use diffusion graph convolution, which was proposed in [20] for its effectiveness in capturing inter-series relations. This formulation captures the relations of node features in K graph convolution iterations. Given node features $X \in \mathbb{R}^{d_{in} \times N}$ and the learned adjacency matrix A, the output node features $Z \in \mathbb{R}^{d_{out} \times N}$ are calculated as follows:

$$Z^\top = \sum_{k=0}^{K} A^k X^\top W^{(k)} \tag{8}$$

where $A^k \in \mathbb{R}^{N \times N}$ is the k-th power of the adjacency matrix A, and $W^{(k)} \in \mathbb{R}^{d_{in} \times d_{out}}$ are learnable parameters at the k-th convolution iteration.

The formula for the graph diffusion convolution in (8) is applied to each step of the i-th ERAN block, given the list of adjacency matrices $A_1, A_2, \ldots, A_L$. In detail, at the i-th ERAN block, the graph diffusion convolution is applied at

each time step as follows.

$$Z_t^\top = \sum_{k=0}^{K} A_t^k X_t^\top W^{(k)} \tag{9}$$

where $Z_t \in \mathbb{R}^{d_{out} \times N}$ is the output node features, and $X_t \in \mathbb{R}^{d_{in} \times N}$ is the input node features at time step t, respectively. It is important to note that at the i-th ERAN block, the length of the temporal model's output is L_i, which is smaller than L. Therefore, only the last L_i adjacency matrices are used.

3.4.3 Residual and skip connections

The classical residual network (ResNet) architecture introduced a skip connection that adds the input tensor to the output tensor of a stack of layers, which is then passed to the next stack [13]. This helps alleviate the problem of vanishing gradients and improves the performance of deep neural networks. In this work, we adopt a similar approach in our proposed model, to enhance the trainability of the model. However, due to the downsampling effect of the TAC module, the length of the residual tensor may be shorter than that of the input tensor. To ensure that the input tensor and the residual tensor have the same dimensions for addition, we truncate the input tensor to match the dimensions of the residual tensor. The addition operation is defined as follows:

$$\mathcal{X}^{(i+1)} = \mathcal{X}^{(i)}[:, :, :, L - L_i : L] + \mathcal{R}^{(i)} \tag{10}$$

with L_i is the length of $\hat{\mathcal{X}}^{(i)}$, the residual output of the i-th ERAN block. Note that, L_i is also the length of the input for the $(i+1)$-th ERAN block.

The skip connection at each consists of a 2D convolution with a kernel size of $(1, L_i)$. The purpose of a skip connection module is to combine all steps of $\mathcal{C}^{(i)}$, the skip output of the i-th TAC, into $C^{(i)} \in \mathbb{R}^{D_c \times N}$, which has a single step:

$$C^{(i)} = \text{Conv2D}\left(\mathcal{C}^{(i)}\right) \tag{11}$$

3.5 Multi-step forecasting

The outputs of the skip connections are summed up and fed to the output layer which generates the prediction of the future $\hat{Y}^f$. The sum of outputs of the skip connections is as follows.

$$C = \sum_{i=1}^{m} C^{(i)} \in \mathbb{R}^{D_c \times N} \tag{12}$$

The output layer consists of two 2D convolutions with a kernel size of (1, 1), which are used to translate the dimension of the input (D_c) to the forecasting horizontal dimension. In other words, the dimensionality of the output layer's input is D_c, while its the dimentionality of its output is τ.

4 Experiments

4.1 Experimental settings

To measure the time-series forecasting performance of ERAN, we use the following public datasets, ranging from traffic flow, electricity consumption, to COVID-19[2]. A summary of these datasets is presented in Table 1.

- *PEMS-03*: contains traffic flow information collected by 358 sensors in the San Francisco Bay Area from Sep 1, 2018, to Nov 30, 2018.
- *PEMS-04*: contains traffic flow information collected by 307 sensors in the San Francisco Bay Area from Jan 1, 2018, to Feb 28, 2018.
- *PEMS-08*: contains traffic flow information collected by 170 sensors in the San Bernardino area from Jul 1, 2016, to Aug 31, 2016.
- *Solar-Energy*: the solar power production records in the year of 2006, sampled every 10 minutes from 137 PV plants in Alabama State.
- *Electricity*: the hourly electricity consumption of 321 clients recorded from 2012 to 2014.
- *COVID-19 Global*: the country-wise daily new cases and daily death tolls of COVID-19 in 25 countries, collected by John Hopkins University.
- *COVID-19 US*: the state-level daily new cases and death tolls of COVID-19 in the US, collected by John Hopkins University.

For PEMS-03, PEMS-04, and PEMS-08, we use a one-hour historical window (12 steps) to predict the values in a 15-minute forecasting horizon (3 steps), following [5]. For Solar, we use a 4-hour historical window (24 steps) to predict the values in the next 0.5 hour forecasting horizon (3 steps). For Electricity, we use a 24-hour historical window (24 steps) to predict the values in the next 3-hour forecasting horizon (3 steps). For COVID-19 Global data, we use a 7-day historical window to predict the daily values in the next 7-day forecasting horizon. For COVID-19 US, we use a historical window of 14 days to predict the values in the next 7-day forecasting horizon. We evaluate the methods by comparing the predicted values with the ground truth using MAE (Mean Absolute Error), RMSE (Root Mean Square Error), and Mean

[2] The datasets used in this paper is available at the following URL: https://drive.google.com/drive/folders/1vsF7dzpiCAKOWpUJw6u1ne1TjMa5m-ZD

Table 1 Summary of the datasets

Dataset	# Nodes	# Time steps	Sampling rate	Predefined graph
PEMS-03	358	26,209	5 min	Available
PEMS-04	307	34,272	5 min	Available
PEMS-08	170	17,856	5 min	Available
Solar	137	52,560	10 min	Not available
Electricity	321	26,304	1 hour	Not available
COVID-19 Global	25	160	1 day	Not available
COVID-19 US	54	320	1 day	Not available

Average Percentage Error (MAPE), which are widely used evaluation metrics in forecasting tasks.

4.2 Baseline methods

To verify the effectiveness of the proposed model, we compare it with state-of-the-art methods for time-series forecasting from the following groups: (i) traditional methods, (ii) deep learning-based methods, (iii) methods that use a pre-defined graph, and (iv) methods that automatically generate a graph from the data. The details of the baselines are as follows.

- *VAR* (Vector Auto-Regression) [15]: An auto-regression model for multivariate time series.
- *LSTM* [14]: A type of recurrent neural network (RNN) architecture designed to capture and remember long-range dependencies in sequential data.
- *GRU* [6]: A recurrent neural network for sequence data that efficiently learns dependencies and avoids gradient issues.
- *TCN* [3]: A model for sequential data using convolutional neural network.
- *LSTNet* [18]: A deep neural network that combines convolutional neural networks and recurrent neural networks.
- *Reformer* [17]: A memory efficient transformer-based model for multivariate time series forecasting.
- *Informer* [38]: A transformer-based model for multivariate time series forecasting.
- *Crossformer* [37]: A transformer-based model for multivariate time series forecasting using cross dimension dependency.
- *DCRNN* [20]: A convolutional recurrent neural network that combines graph convolutions with recurrent neural networks.
- *ST-GCN* [36]: A spatial-temporal graph convolutional network that combines a graph convolution with 1D convolutions.
- *GWNet* (Graph Wavenet) [34]: A spatial-temporal graph convolutional network that combines graph convolutions with 1D dilated convolutions.
- *MT-GNN* [35]: A method that learns to generate a static graph from the data and then combines graph convolutions with 1D convolutions.
- *StemGNN* [5]: A method that learns to generate a static graph from the data and uses graph neural networks and 1D convolutions in the spectral domain.
- *AGCRN* [2]: A method that learns to generate a static graph from the data and combines graph convolutions with recurrent neural networks.

4.3 Implementation details

In our proposed model, ERAN, we utilized a three-layer architecture with input and output dimensionalities of 128. The selection of window size and dilation factor was contingent upon the length of the historical window. In the case of a long historical window, our goal is to have an expansive receptive field that covers the entire sequence. To achieve this, we utilize a long window size and a large dilation factor. In contrast, for a shorter historical window, a smaller window size and dilation factor are sufficient. In detail, for data with length smaller than or equal to 12 steps, we used a window size of 3 and dilation of 1, while for data with length greater than 12 steps, we used a window size of 5 and dilation factor of 2. We employed the Adam optimiser with a learning rate of 0.001 and weight decay of 0.0001.

All deep learning-based models were implemented using PyTorch and trained on a machine equipped with a single NVIDIA GPU. We halted training after 100 epochs and reported results on the test set for the epochs that produced the lowest loss on the validation set.

4.4 Results

4.4.1 Overall comparison

Tables 2, 3 and 4a provide a comprehensive comparison of the methods across the datasets. The results show that ERAN outperforms all competing methods on all datasets. In addition, we make the following observations:

1. In general, deep learning-based methods perform better than traditional methods, except for VAR, which performs comparably to some deep learning-based methods on certain traffic flow datasets.

Table 2 Results on traffic flow forecasting (historical window: 12 steps, forecast horizon: 3 steps)

	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
	PEMS-03			PEMS-04			PEMS-08		
VAR	23.65	38.26	0.245	24.54	38.61	0.172	19.19	29.81	0.131
LSTM	21.33	35.11	0.233	27.14	41.59	0.182	22.20	34.06	0.142
GRU	21.18	34.84	0.183	27.41	40.34	0.179	22.43	34.31	0.145
TCN	18.23	25.04	0.194	26.31	36.11	0.156	15.93	25.69	0.165
LSTNet	19.07	29.67	0.177	24.04	37.38	0.170	20.26	31.96	0.113
Reformer	17.35	30.72	0.174	21.09	33.95	0.151	19.17	29.18	0.126
Informer	18.47	32.41	0.182	22.17	36.00	0.173	20.41	31.95	0.125
Crossformer	15.69	24.76	0.164	20.84	32.12	0.150	16.26	24.78	0.125
DCRNN	18.18	30.31	0.181	19.54	31.22	0.171	17.86	27.83	0.114
ST-GCN	17.49	30.12	0.171	20.03	33.21	0.145	18.02	27.83	0.114
GWNet	19.85	32.94	0.193	26.85	39.70	0.172	19.13	28.16	0.126
StemGNN	14.32	21.64	0.162	20.24	32.15	**0.101**	15.83	24.93	0.093
MTGNN	14.27	21.13	0.155	20.12	30.47	0.122	15.04	22.45	0.111
AGCRN	14.46	21.97	0.156	19.12	30.98	0.131	15.95	25.22	0.112
ERAN (ours)	**13.89**	**20.94**	**0.148**	**18.76**	**29.48**	0.104	**14.46**	**22.26**	**0.091**

2. Among deep learning-based methods, graph-based models outperform non-graph-based models (such as LSTM, TCN, and LSTNet), highlighting the significance of explicitly modeling inter-series dependencies.
3. Our proposed model, ERAN, outperforms other methods, that model the inter-channel relations, by a significant margin. The key difference is that while these methods only capture static inter-channel relations, ERAN models the evolution of such relations via an evolving graphs and thus captures dynamic dependencies between variables more effectively.

Table 3 Results on the energy consumption forecasting (historical window: 24 steps, forecast horizon: 3 steps)

	MAE	RMSE	MAPE	MAE	RMSE	MAPE
	Solar-Energy			Electricity		
VAR	2.27	3.25	0.925	268.4	1811.3	0.194
LSTM	2.18	3.17	0.846	230.3	1725.4	0.246
GRU	1.98	3.05	0.878	230.3	1725.4	0.225
TCN	2.05	3.09	0.735	221.1	1698.5	0.182
LSTNet	2.15	3.06	0.684	248.9	1625.3	0.178
Reformer	1.43	2.41	0.751	208.7	1183.4	0.197
Informer	1.57	2.58	0.786	215.5	1296.8	0.199
Crossformer	1.52	2.65	0.676	336.7	2408.9	0.205
StemGNN	1.19	2.26	0.651	336.7	2408.9	0.244
GWNet	2.75	4.05	0.977	336.7	2408.9	0.407
MTGNN	1.19	2.23	0.452	204.2	1645.4	0.133
AGCRN	1.15	2.21	0.519	201.5	1308.7	0.224
ERAN (ours)	**1.08**	**2.05**	**0.446**	**170.9**	**1107.4**	**0.119**

Due to the unavailability of pre-defined graphs, models that are dependent on pre-defined graphs are excluded

To evaluate the long-term forecasting ability, we report the mean absolute error (MAE) of the predictions at different forecasting horizons for the PEMS-04 and Electricity datasets in Fig. 3. Our results demonstrate that ERAN outperforms all competing methods across all forecasting horizons, showing its effectiveness in predicting long-term trends. Particularly, for the longest horizon (60 minutes), the difference between ERAN and the other methods is significant, emphasizing ERAN's superior long-term forecasting ability.

4.4.2 Significance testing

To assess the statistical significance of our results, we conducted pairwise hypothesis tests comparing the absolute errors generated by the ERAN against those produced by the second-best model. The hypotheses were formulated as follows: H_0 (null hypothesis) posits that there is no difference between the mean absolute errors (MAE) of the two algorithms, while H_a (alternative hypothesis) asserts that the mean absolute errors of the two algorithms are statistically different.

The significance level, denoted by a threshold of 0.05 (corresponding to a confidence level of 95%), was used to assess the p-values. If the p-value is smaller than this threshold, the null hypothesis is rejected.

The t-test results are outlined in Table 5. Notably, all pairwise tests exhibit p-values significantly below 0.05. This lack of evidence supports the rejection of the null hypothesis,

Table 4 Results on COVID-19 new cases and death tolls forecasting (historical window: 7 steps, forecast horizon: 7 steps)

	MAE	RMSE	MAPE	MAE	RMSE	MAPE
	Global new cases			Global death toll		
(a) Results on global new cases and death tolls forecasting (historical window: 7 steps, forecast horizon: 7 steps).						
VAR	2,211	3,847	3.621	178.51	235.42	6.428
LSTM	1,783	3,421	3.586	172.13	211.61	6.132
GRU	1,761	3,215	3.421	173.21	214.35	6.241
TCN	1,787	3,414	3.472	175.32	223.73	6.311
LSTNet	1,761	3,312	3.045	167.45	198.56	5.729
Reformer	3,305	6,728	2.501	228.91	361.25	6.386
Informer	3,608	7,562	3.162	171.45	281.94	7.109
Crossformer	2,050	5,573	3.515	116.50	312.50	3.717
StemGNN	2,828	7,186	1.854	72.32	161.45	5.345
GWNet	1,340	3,796	1.306	96.54	152.76	4.921
MTGNN	1,365	3,294	1.528	73.05	151.92	4.955
AGCRN	1,396	3,417	1.222	116.40	296.29	7.354
ERAN (ours)	**1,282**	**3,027**	**0.873**	**69.93**	**143.21**	**4.811**
	US new cases			US death toll		
(b) Results on US new cases and death tolls forecasting (historical window: 14 steps, forecast horizon: 7 steps).						
VAR	1,938	4,821	1.632	41.13	78.61	1.428
LSTM	1,808	4,905	1.268	39.43	76.82	1.032
GRU	1,791	4,862	1.127	39.28	75.93	1.004
TCN	1,754	4,899	1.213	38.45	72.58	1.341
LSTNet	1,821	3,726	1.028	40.21	78.32	1.213
Reformer	3,149	6,338	1.067	45.58	87.39	1.025
Informer	2,661	5,559	2.231	39.13	76.64	1.001
Crossformer	1,650	4,547	1.338	29.59	61.58	1.282
StemGNN	2,419	5,375	1.361	35.53	64.85	1.042
GWNet	1,446	4,165	1.052	28.73	50.28	1.015
MTGNN	1,458	3,981	1.015	28.32	48.87	1.016
AGCRN	1,581	4,929	1.005	37.03	70.13	1.311
ERAN (ours)	**1,060**	**2,348**	**0.781**	**24.15**	**45.64**	**0.979**

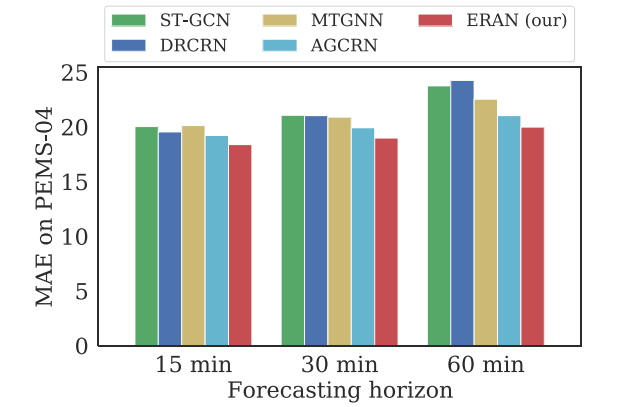
Due to the unavailability of pre-defined graphs, models that are dependent on pre-defined graphs are excluded

signifying a statistically significant difference between the ERAN MAE and that of the next best model.

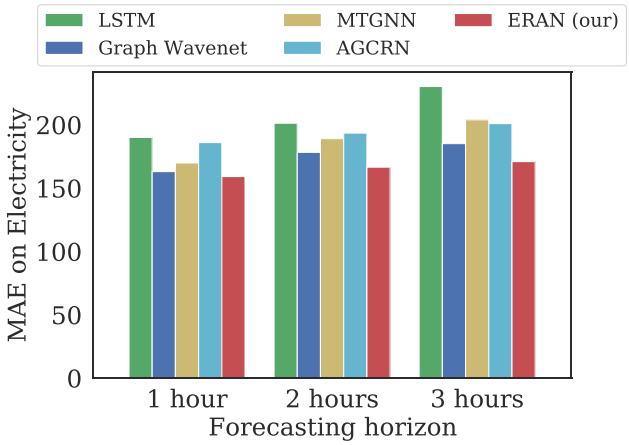
4.4.3 Impact on non-stationary time series

To evaluate the efficacy of ERAN on non-stationary time series, specifically, we investigate the stationarity of time series and compare how ERAN outperforms existing models in both stationary and non-stationary contexts.

We begin by examining the stationarity of time series using the Augmented Dickey-Fuller test (ADF) [9]. The ADF test is a widely employed statistical method for determining whether a given time series is stationary. The test formulates a null hypothesis assuming the presence of a unit root, indicating non-stationarity, and an alternative hypothesis suggesting stationarity. The test statistic is then compared to critical values, and the resulting p-value is pivotal in establishing the stationarity of the time series. A p-value below a chosen significance level (commonly 0.05) leads to the rejection of the null hypothesis, providing evidence in favor of stationarity. In contrast, a p-value exceeding the significance level leads to the acceptance of the null hypothesis, implying the presence of non-stationarity in the time series. Therefore, the ADF test utilizes the p-value to make informed decisions about the stationarity of the analysed time series. The results of the ADF test regarding the stationarity of time series are presented in Table 6.



(a) MAE on PEMS-04



(b) Electricity

Fig. 3 MAE at different forecasting horizons

Next, we assess the enhancement in Mean Absolute Error (MAE) of ERAN compared to the second-best model for each dataset. From the outcomes presented in Table 7, we can see that the enhancements on non-stationary time series are more substantial than those on stationary ones. Across five stationary time series, the average percentage decrease of MAE is 5.93%, whereas over four non-stationary time series, the average percentage decrease of MAE is 12.25%. These results confirm the effectiveness of ERAN in enhancing forecast accuracy, particularly in the context of non-stationary time series.

Table 5 Significance testing results between ERAN and the Next best model for each dataset

Dataset	Next best model	p-value	Diff?
PEMS-03	MTGNN	7.15e-06	Yes
PEMS-04	AGCRN	3.42e-06	Yes
PEMS-08	MTGNN	2.51e-05	Yes
Solar	AGCRN	2.45e-05	Yes
Electricity	AGCRN	1.66e-05	Yes
COVID-19 Global new cases	GWNet	5.76e-05	Yes
COVID-19 Global death toll	StemGNN	1.23e-05	Yes
COVID-19 US new cases	GWNet	1.47e-05	Yes
COVID-19 US death toll	MTGNN	2.63e-05	Yes

Table 6 The results of ADF test

Dataset	p-value	Stationary
PEMS-03	0.0	Yes
PEMS-04	1.11e-25	Yes
PEMS-08	2.32e-26	Yes
Solar	0.0	Yes
Electricity	0.0011	Yes
COVID-19 Global new cases	0.6939	No
COVID-19 Global death toll	0.9989	No
COVID-19 US new cases	1.0	No
COVID-19 US death toll	0.9983	No

A p-value greater than 0.05 indicates that the time series is non-stationary

4.5 Ablation study

To gain more insight into the proposed model, we conducted an ablation study to evaluate the impact of (i) learning the dynamics of the inter-channel relations, and (ii) learning the dynamics of the intra-channel relations; (iii) the number of the layers; (iv) the dilation factor, on the model's performance as follows.

Table 7 MAE Improvement of ERAN relative to the second best model across datasets

Dataset	Stationary	Second best model	% MAE decrease
PEMS-03	Yes	MTGNN	2.66%
PEMS-04	Yes	AGCRN	1.88%
PEMS-08	Yes	MTGNN	3.85%
Solar	Yes	AGCRN	6.08%
Electricity	Yes	AGCRN	15.18%
COVID-19 Global new cases	No	GWNet	4.32%
COVID-19 Global death toll	No	StemGNN	3.30%
COVID-19 US new cases	No	GWNet	26.69%
COVID-19 US death toll	No	MTGNN	14.72%

The findings indicate a more pronounced enhancement in MAE for non-stationary time series compared to stationary counterparts

Table 8 Ablation study on the impact of the graphs

	MAE	RMSE	MAE	RMSE
	Global death toll		US death toll	
Evolving graph	**69.93**	**143.21**	**24.15**	**45.64**
Static graph	71.28	146.35	26.18	47.21
No graph	72.25	148.98	26.51	47.54

4.5.1 Impact of learning the dynamics of inter-channel relations

To evaluate the effectiveness of learning the dynamics of inter-channel relations, we examined three variants of ERAN: (1) *Evolving graph*: where the inter-channel relations is modelled by an an elvoving graph generated by the EGL layer, (2) *Static graph*: where the inter-channel relations is modelled by a static graph generated from the data, which is unchanged thoughout the time series lifetime, and (3) *No graph*: where inter-channel relations are ignored.

Table 8 reports the results on the COVID-19 dataset. We observe that the *Evolving graph* variant, which uses the evolving graph, performs better than the other variants, demonstrating the importance of capturing the evolution of inter-channel relations over time. Additionally, we see that the *Static graph* variant, which can only capture a static relation between the variables, improves the forecasting accuracy. Finally, the *No graph* variant performs the worst among the three variants, emphasizing the effectiveness of capturing the relations between the variables using graph convolution.

4.5.2 Impact of learning the dynamics of the intra-channel relations

As the learning the dynamics of intra-channel relations is accomplished through the TAC module, we investigated the impact of this learning approach by comparing TAC with LSTM and TCN, which capture invariant temporal patterns only. It's important to note that TAC is achieved by removing the graph from ERAN, making it equivalent to ERAN's "no graph" variant presented in Section 4.5.1.

The results in Table 9 show that TAC significantly outperforms LSTM and TCN on COVID transmission data, a highly non-stationary time series. This indicates the effectiveness of using TAC in capturing the dynamics of the temporal patterns.

Table 9 Study on the impact of TAC

	MAE	RMSE	MAE	RMSE
	Global death toll		US death toll	
TAC	**72.25**	**148.98**	**26.51**	**47.54**
LSTM	172.13	211.61	39.43	76.82
TCN	175.32	223.73	38.45	72.58

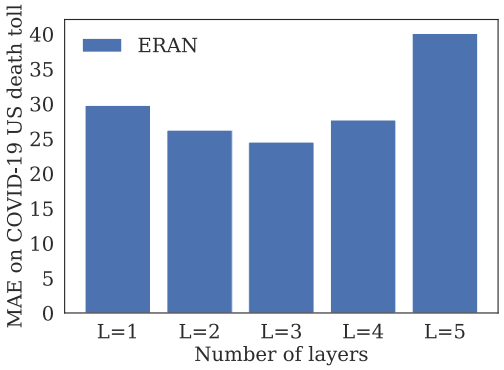


Fig. 4 MAE on COVID-19 US daily death toll with different number of layers

4.5.3 Impact of the number of layers

One important parameter of the ERAN is the number of layers. To demonstrate the impact of the number of layers, we present a plot of the MAE on COVID-19 US death tolls in Fig. 4. The optimal forecasting accuracy is achieved when the number of layers is 3. With a small number of layers, the model's capacity is too limited to learn the data. On the other hand, if we increase the number of layers, the model capacity will increase and is prone to over-fitting.

4.5.4 Impact of the dilation factor

We use a dilation factor p to control the dilation at each layer. The dilation of the next layer is p times the dilation of the previous one. Thus, the dilation of layer l is p^{l-1}. Table 10 shows the forecasting accuracy of COVID-19 in the US using dilation factors 1 and 2. We can see that dilation factor 2 achieves slightly better forecasting accuracy than dilation factor 1. This confirms the effectiveness of using a dilation factor larger than 1 to extend the receptive field.

4.6 Qualitative analysis

To gain more insight into the behavior of the methods, we plot in Fig. 5 two examples of the daily new cases forecasting

Table 10 Ablation study on the impact of the dilation factor

	MAE	RMSE	MAE	RMSE
	US new cases		US death toll	
p=2	**1,060**	**2,348**	**24.15**	**45.64**
p=1	1,097	2,389	26.11	47.53

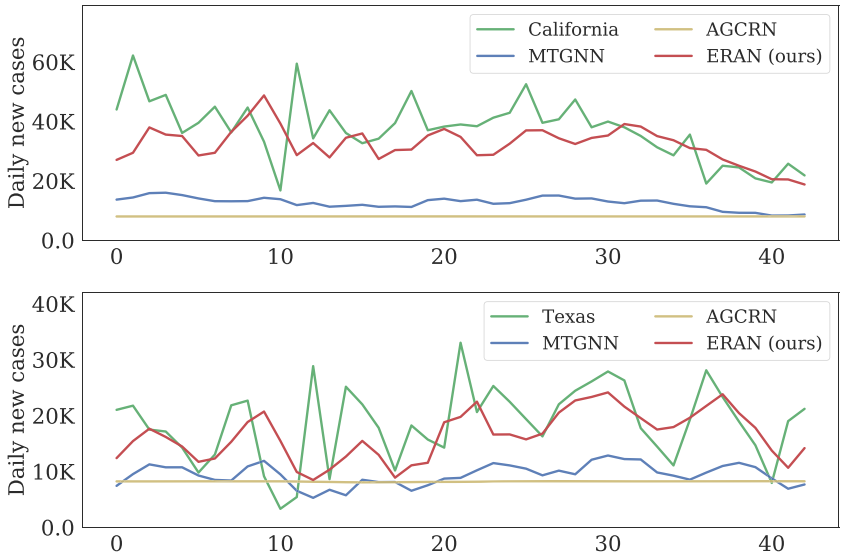


Fig. 5 Visualisation of COVID-19 forecasting

for COVID-19 in the US, 7 days in advance. The blue line represents the ground truth data for each state, while the other colors represent the forecasting results of the models. Overall, in both examples, ERAN performed better than other methods in predicting the ground truth data. ERAN was also better in capturing the trend when the ground truth went up and down. However, in both cases, there were some short but sharp spikes that no method could capture well.

5 Conclusion

We propose ERAN, a model that captures the dynamics of the intra- and inter-channel relations for multivairate time series forecasting. To model the intra-channel relations, ERAN utilises Temporal Attentional Convolution (TAC), which applies self-attention mechanism within a temporal window. On the other hand, to model the inter-channel relations, ERAN uses dynamic graph convolutional network, wherein the graph structure evolves over time. Our experimental architecture has established new state-of-the-art results on multiple types of time series data, from classical traffic flows and electricity consumption forecasting to newly emerging problems like COVID- 19 projections. Furthermore, ERAN exhibits significant improvement over existing methods, particularly evident in non-stationary time series. The representation power and generality of the model promise strong and wide applications in time series modeling. However, a notable limitation of the proposed model is the current exclusion of time-dependent covariates such as weather and price indices as inputs for forecasting. Recognizing this, we identify the incorporation of these covariates as a potential avenue for improvement, which we leave for future work.

Acknowledgements This work is partially funded by the UK Research and Innovation (UKRI) as a Knowledge Transfer Partnership (KTP) under project number 12493.

Author Contributions Binh Nguyen-Thai proposed the research idea, designed and implemented the model, conducted experiments, and served as the primary writer. Vuong Le, Ngoc-Dung T. Tieu, and Truyen Tran contributed to the model design, result analysis, and manuscript revisions. Svetha Venkatesh, and Naeem Ramzan supervised and participated in revising the paper.

Data Availability The datasets used in this paper is available at the following URL: https://drive.google.com/drive/folders/1vsF7dzpiCAKOWpUJw6u1ne1TjMa5m-ZD

Declarations

Ethical standard All datasets used in this paper are publicly available and no consent was required for their use.

Competing Interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copy-

right holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Alexandrov A, Benidis K, Bohlke-Schneider M et al (2020) Gluonts: probabilistic and neural time series modeling in python. J Mach Learn Res 21(116):1–6. http://jmlr.org/papers/v21/19-820.html
2. Bai L, Yao L, Li C et al (2020) Adaptive graph convolutional recurrent network for traffic forecasting. In: Larochelle H, Ranzato M, Hadsell R et al (eds) Advances in neural information processing systems, vol 33. Curran Associates, Inc., pp 17,804–17,815
3. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271
4. Bruna J, Zaremba W, Szlam A et al (2014) Spectral networks and locally connected networks on graphs. International conference on learning representations (ICLR 2014)
5. Cao D, Wang Y, Duan J et al (2020) Spectral temporal graph neural network for multivariate time-series forecasting. Adv Neural Inf Process Syst 33
6. Cho K, van Merriënboer B, Gulcehre C et al (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1724–1734
7. Conti M, Turchetti C (1994) Approximation of dynamical systems by continuous-time recurrent approximate identity neural networks. Neural Parallel Sci Comput 2(3):299–320
8. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th international conference on neural information processing systems, NIPS'16, pp 3844–3852
9. Dickey DA, Fuller WA (1979) Distribution of the estimators for autoregressive time series with a unit root. J Am Stat Assoc 74(366):427–431
10. Dudek G (2015) Short-term load forecasting using random forests. In: Filev D, Jabłkowski J, Kacprzyk J et al (eds) Intelligent systems'2014. Springer International Publishing, Cham, pp 821–828
11. Gilmer J, Schoenholz SS, Riley PF et al (2017) Neural message passing for quantum chemistry. In: Proceedings of the 34th international conference on machine learning, ICML'17, vol 70. pp 1263–1272
12. Guo S, Lin Y, Feng N et al (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: Proceedings of the AAAI conference on artificial intelligence, pp 922–929
13. He K, Zhang X, Ren S et al (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778. https://doi.org/10.1109/CVPR.2016.90
14. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
15. Hyndman R, Athanasopoulos G (2021) Forecasting: principles and practice, 3rd edn. OTexts, Australia
16. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations, ICLR '17
17. Kitaev N, Kaiser L, Levskaya A (2020) Reformer: the efficient transformer. In: International conference on learning representations. https://openreview.net/forum?id=rkgNKkHtvB
18. Lai G, Chang WC, Yang Y et al (2018) Modeling long- and short-term temporal patterns with deep neural networks. In: The 41st international ACM SIGIR conference on research and development in information retrieval, SIGIR'18. New York, pp 95–104
19. Li S, Jin X, Xuan Y et al (2019) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Advances in neural information processing systems, pp 5243–5253
20. Li Y, Yu R, Shahabi C et al (2018) Diffusion convolutional recurrent neural network: data-driven traffic forecasting. In: International conference on learning representations (ICLR '18)
21. Lim B, Arık SÖ, Loeff N et al (2021) Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecasting 37(4):1748–1764
22. Mills TC (1990) Time series techniques for economists. Cambridge University Press
23. Nguyen D, Nguyen B, Nguyen P et al (2021) High-order representation learning for multivariate time series forecasting. In: Time series workshop@ICML 2021
24. Oord Avd, Dieleman S, Zen H et al (2016) Wavenet: a generative model for raw audio. arXiv:1609.03499
25. Pai PF, Lin KP, Lin CS et al (2010) Time series forecasting by a seasonal support vector regression model. Expert Syst Appl 37(6):4261–4265. https://doi.org/10.1016/j.eswa.2009.11.076, https://www.sciencedirect.com/science/article/pii/S0957417409010185
26. Pareja A, Domeniconi G, Chen J et al (2020) EvolveGCN: evolving graph convolutional networks for dynamic graphs. In: Proceedings of the thirty-fourth AAAI conference on artificial intelligence
27. Pham T, Tran T, Phung D et al (2017) Column networks for collective classification. In: Proceedings of AAAI conference on artificial intelligence
28. Rangapuram SS, Seeger MW, Gasthaus J et al (2018) Deep state space models for time series forecasting. In: Bengio S, Wallach H, Larochelle H et al (eds) Advances in neural information processing systems, vol 31. Curran Associates, Inc., https://proceedings.neurips.cc/paper_files/paper/2018file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf
29. Rasul K, Seward C, Schuster I et al (2021) Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In: International conference on machine learning. https://api.semanticscholar.org/CorpusID:231719657
30. Salinas D, Flunkert V, Gasthaus J et al (2020) Deepar: probabilistic forecasting with autoregressive recurrent networks. Int J Forecasting 36(3):1181–1191. https://www.sciencedirect.com/science/article/pii/S0169207019301888
31. Shaw P, Uszkoreit J, Vaswani A (2018) Self-attention with relative position representations. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, vol 2 (Short Papers). Association for Computational Linguistics, New Orleans, pp 464–468. https://doi.org/10.18653/v1/N18-2074, https://aclanthology.org/N18-2074
32. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. In: Proceedings of the 31st international conference on neural information processing systems, NIPS'17, pp 6000–6010
33. Wu N, Green B, Ben X et al (2020a) Deep transformer models for time series forecasting: the influenza prevalence case. arXiv:2001.08317
34. Wu Z, Pan S, Long G et al (2019) Graph WaveNet for deep spatial-temporal graph modeling. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19.

International Joint Conferences on Artificial Intelligence Organization, pp 1907–1913
35. Wu Z, Pan S, Long G et al (2020b) Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, pp 753–763
36. Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Proceedings of the 27th international joint conference on artificial intelligence (IJCAI)
37. Zhang Y, Yan J (2023) Crossformer: transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: International conference on learning representations
38. Zhou H, Zhang S, Peng J et al (2021) Informer: beyond efficient transformer for long sequence time-series forecasting. In: The thirty-fifth AAAI conference on artificial intelligence, AAAI 2021. AAAI Press, p online

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Binh Nguyen-Thai[1,2] · Vuong Le[3] · Ngoc-Dung T. Tieu[4,5] · Truyen Tran[6] · Svetha Venkatesh[6] · Naeem Ramzan[1]

Vuong Le
levuong@amazon.com

Ngoc-Dung T. Tieu
tieungocdung@gmail.com

Truyen Tran
truyen.tran@deakin.edu.au

Svetha Venkatesh
svetha.venkatesh@deakin.edu.au

Naeem Ramzan
naeem.ramzan@uws.ac.uk

1 School of Computing, Engineering and Physical Sciences, University of the West of Scotland, High St, Paisley PA1 2BE, UK

2 Raven Controls Limited, 227 West George Street, Glasgow G2 2ND, UK

3 Amazon, 555 Collins street, Melbourne, Victoria 3000, Australia

4 Faculty of Information Technology, University of Transport and Communications, 3 Cau Giay, Dong Da, Hanoi, Vietnam

5 University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

6 Applied AI Institute, Deakin University, 75 Pigdons Road, Waurn Ponds, VIC 3126, Australia