Evolutionary Neural Architecture Search for Multivariate Time Series Forecasting

Zixuan Liang LIANGZIXUAN@STU.SCU.EDU.CN
College of Computer Science, Sichuan University.
Yanan Sun* YSUN@SCU.EDU.CN
College of Computer Science, Sichuan University.

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

Multivariate time series forecasting is of great importance in a diverse range of domains. In recent years, a variety of spatial-temporal graph neural networks (STGNNs) have been proposed to address this task and achieved promising results. However, these networks are typically handcrafted and require extensive human expertise. Additionally, the temporal and spatial dependencies hidden within different scenarios vary, making it difficult for them to adapt to different scenarios. In this paper, we propose an evolutionary neural architecture search framework, entitled EMTSF, for automated STGNN design. Specifically, we employ fine-grained neural architecture search into both the spatial convolution module and the temporal convolution module. For the spatial convolution search space, various feature filtering and neighbor aggregation operations are employed to find the most suitable message-passing mechanism for modeling the spatial dependencies. For the temporal convolution search space, gated temporal convolutions with different kernel sizes are utilized to best capture temporal dependencies with various ranges. The spatial convolution module and temporal convolution module are jointly optimized with the proposed evolutionary search algorithm to heuristically identify the optimal STGNN architecture. Extensive experiments on four commonly used benchmark datasets show EMTSF achieves promising performance compared with the state-of-the-art methods, which demonstrates the effectiveness of the proposed framework.

Keywords: Multivariate time series forecasting, Neural architecture search, Deep learning.

1. Introduction

With the development of sensors and communication technology, time series data is becoming ubiquitous. Accurate time series forecasting enables decision-making in many challenging domains, e.g., transportation Li et al. (2018), power management Simeunović et al. (2021), and health care Jin et al. (2018). In contrast to univariate time series forecasting, where only one time series is considered, multivariate time series forecasting takes into account the relationships between a group of time series to produce more accurate predictions. Hence, multivariate time series forecasting methods have attracted increasing interest among researchers.

* Corresponding author

To create an effective multivariate time series forecasting model, it is essential to accurately capture the spatial correlation among different time series and the temporal dependence within a single time series. In recent years, the rapid advancement of deep learning techniques has led to the emergence of several deep learning-based methods for multivariate time series forecasting. Two of the pioneering works in this area are LSTNet Lai et al. (2018) and TPA-LSTM Shih et al. (2019), which combine the convolutional neural network (CNN) and recurrent neural network (RNN) to capture the spatial correlation and temporal dependence, respectively. However, the global aggregation of CNN makes it challenging to capture pair-wise correlations among time series precisely Ye et al. (2022). In fact, multivariate time series data can be viewed as a graph, where each node corresponds to a time series, and the edge describes the pair-wise correlation between time series. Therefore, leveraging graph neural networks (GNNs) Atwood and Towsley (2016); Hamilton et al. (2017) to capture the interdependencies among time series holds great promise.

Recently, a growing number of spatial-temporal graph neural networks (STGNNs) Yu et al. (2017); Li et al. (2018); Ye et al. (2022) have been proposed for multivariate time series forecasting. STGNNs use GNNs for spatial modeling, along with employing temporal convolutional networks (TCNs) Bai et al. (2018) or RNNs for temporal modeling, which have become the mainstream method for multivariate time series forecasting. Despite their remarkable success, most existing STGNNs are still manually designed. This manual design process requires significant human effort and rich domain knowledge, which is not readily available to most interested users. Moreover, neural architectures are usually tailored to specific problems, while the spatial correlations and temporal dependencies of data may vary across different problem scenarios, necessitating a redesign of the architecture to meet performance requirements. Consequently, there is a growing demand for an automated neural architecture design framework for multivariate time series forecasting.

Fortunately, neural architecture search (NAS) Elsken et al. (2019) has emerged as a promising approach to automate architecture design without manual intervention. There are two most critical components in NAS: (1) search space and (2) search strategy. The search space refers to the set of all possible neural network architectures generated during the search process. A well-defined search space can help to identify novel architectures with high performance. The search strategy details how to explore the search space. A good search strategy can improve search efficiency and effectiveness. NAS has been successfully applied to various fields, including image classification Sun et al. (2020), segmentation Liu et al. (2019a), and object detection Chen et al. (2019). AutoSTG Pan et al. (2021) first extended NAS to the urban traffic prediction field and proposed a NAS framework for automated STGNN design. However, they adopt pre-defined and fixed components as the available operations in their search space. It is essentially an ensemble and fine-tuning of the existing STGNNs, leading to redundant computation and limiting the potential for capturing spatial correlations and temporal dependencies. Furthermore, they utilize the gradient-based method Liu et al. (2019b) as their search strategy. Despite the high computational efficiency, it suffers from excessive memory consumption. When dealing with large, complex, multivariate time series datasets, adopting gradient-based methods to explore the fine-grained search space can lead to out-of-memory issues.

In this paper, we propose a novel evolutionary neural architecture search framework, entitled EMTSF, for multivariate time series forecasting. Specifically, we carefully designed

a fine-grained search space for the spatial convolution module and the temporal convolution module of STGNN. For the spatial convolution search space, various feature filtering and neighbor aggregation operations are included for finding a powerful GNN with the most suitable message-passing mechanism, which can effectively capture the intricate spatial correlation among time series. For the temporal convolution search space, the combinations of temporal convolution filters with different filter sizes are explored to discover a gated TCN, which can accurately capture the complex temporal dependence across different ranges. Based on the designed search space, we further propose an evolutionary search algorithm with a well-designed encoding strategy and genetic operation to heuristically discover the optimal spatial and temporal convolution modules, thereby reducing human intervention during architecture design and enabling us to better explore the merits of STGNN in multivariate time series forecasting. Our major contributions are outlined as follows:

- We design a fine-grained STGNN search space for multivariate time series forecasting, which allows us to discover powerful GNNs with novel message-passing mechanisms for spatial correlation modeling, as well as identify the combinations of temporal convolution filters to form TCNs that can accurately capture temporal dependencies.
- We propose an evolutionary search algorithm to automate neural architecture design for multivariate time series forecasting. Specifically, a well-designed encoding strategy is proposed to encode the architecture of STGNN, and effective genetic operations are proposed to explore the designed search space to identify the optimal neural architecture.
- We conduct extensive experiments on four real-world datasets to verify the effectiveness of the proposed EMTSF approach. The experimental results demonstrate that EMTSF can effectively automate STGNN design and achieve promising performance compared with state-of-the-art baselines.

2. Related Works

2.1. Multivariate Time Series Forecasting

Multivariate time series forecasting plays an important role in a wide range of real-world applications Zhang et al. (2017); Jin et al. (2018); Simeunović et al. (2021). In the early stage, the majority of research in this field employed statistical approaches Box et al. (2015); Frigola (2015), such as the auto-regressive model (VAR) and Gaussian process model (GP). Recent years have witnessed the rapid development of deep learning, and several deep learning-based methods have been proposed for multivariate time series forecasting. For instance, LSTNet Lai et al. (2018) and TPA-LSTM Shih et al. (2019) combined CNN with RNN (such as GRU and LSTM) for modeling the spatial and temporal dependencies, respectively. STGCN Yu et al. (2017) and DCRNN Li et al. (2018) first integrate GNN with temporal modeling techniques and propose STGNNs. Graph-WaveNet Wu et al. (2019) proposed a graph convolution layer with a self-adaptive adjacency matrix to improve the limited spatial relations from the pre-defined traffic networks. Based on this, MTGNN Wu et al. (2020) further proposed a graph learning layer to handle multivariate time series

without an externally pre-defined graph structure, and employs mix-hop graph convolution and dilated inception convolution to capture the spatial and temporal dependencies.

2.2. Neural Architecture search

Neural Architecture Search (NAS) Elsken et al. (2019) is a technique for automating neural architecture design. It involves searching through a vast search space to find an optimal architecture for a given task. Depending on the adopted search strategy, NAS methods can be mainly categorized into three groups, i.e., reinforcement learning (RL)-based NAS methods Zoph and Le (2016); Zoph et al. (2018), gradient-based NAS methods Liu et al. (2019b); Xu et al. (2019), and evolutionary algorithm (EA)-based NAS methods Real et al. (2019); Termritthikun et al. (2021). Generally speaking, RL-based NAS methods often require a significant amount of computational resources Real et al. (2017); Zoph et al. (2018), which limits their practical use. Gradient-based NAS methods have merit in computation efficiency, but they incur high GPU memory usage since all candidate operations in their supernet need to be computed during searching Ren et al. (2021), making it challenging to scale to larger search spaces. Compared with these two methods, the EA-based NAS methods can effectively handle large search spaces without consuming too many computational resources Liu et al. (2021).

3. METHODOLOGIES

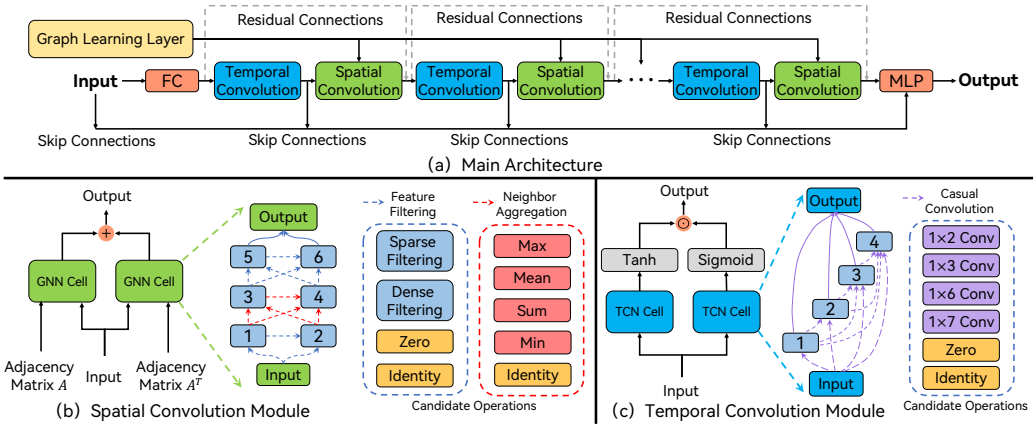


Figure 1: The overview of EMTSF. Figure 1(a) depicts the main architecture of EMTSF. Figure 1(b) and Figure 1(c) provide the detailed structure and search space of the spatial convolution module and temporal convolution module, respectively.

We begin by elaborating on the overall architecture of the proposed EMTSF. As depicted in Figure 1(a), the input is initially projected into a latent space using a fully connected layer. The backbone of the architecture is comprised of interleaved temporal convolution modules and spatial convolution modules, which will be automatically designed to capture temporal dependencies and spatial correlations. To adaptively derive a graph adjacency matrix from the multivariate time series data, we employ a graph learning layer from MTGNN.

This matrix serves as one of the inputs for the spatial convolution modules. Additionally, to mitigate the gradient vanishing problem, residual and skip connections are incorporated. Finally, a multilayer perception is utilized to project the hidden features to the desired output dimension, resulting in the final outputs. In the subsequent subsections, we will briefly describe the search space for the spatial convolution module and the temporal convolution module. Furthermore, we will introduce the evolutionary search algorithm employed in EMTSF to explore the designed search space and discover promising architectures.

3.1. Spatial Convolution Module

GNNs are powerful tools for modeling the complex relationships between entities in graph-structured data. Current GNNs are constructed on the message-passing mechanism, where each node collects information from its neighbors and update its own state. Cai et al. (2021) deconstruct the message-passing GNN and propose a graph neural architecture paradigm that could flexibly combine feature filtering and neighbor aggregation operations to theoretically approximate existing popular GNNs. To explore powerful spatial convolution modules for multivariate time series forecasting tasks, we adopt feature filtering and neighbor aggregation operations as the candidate operations in the search space of spatial convolution modules.

As shown in Figure 1(b), the search space can be viewed as a directed acyclic graph (DAG) consisting of N_s nodes. Each node represents a latent representation and has a directed edge associated with a specific operation that transforms the latent representation. The search space is divided into three parts, each containing one-third of N_s ordered sequence nodes. The first part comprises nodes associated with feature filtering operations that re-scale the information which will be sent to neighbors later. The second part includes nodes associated with neighbor aggregation operations responsible for aggregating information from neighbor nodes. Finally, the third part contains nodes associated with feature filtering operations that preserve critical information received from neighbors.

Feature Filtering. The essence of the feature filtering operation is a gating mechanism for adaptive node feature selection. We employ sparse filtering $F_{sparse}(\cdot)$ and dense filtering $F_{dense}(\cdot)$ as candidate feature filtering operations to re-scale node features at different granularities. These two operations can be expressed as follows:

$$F_{sparse}(H) = Z_s H, \tag{1}$$

$$F_{dense}(H) = Z_d \odot H, \tag{2}$$

where $H \in \mathbb{R}^{n \times d}$ denotes the node embedding, $\odot$ denotes the Hadamard product. $Z_s \in \mathbb{R}^{n \times n}$ and $Z_d \in \mathbb{R}^{n \times d}$ represent the re-scaling matrix calculated jointly from H and H_{in}:

$$Z_s = diag(\sigma(fc_s([H, H_{in}]))), \tag{3}$$

$$Z_d = \sigma(fc_d([H, H_{in}])), \tag{4}$$

where fc_s, fc_d denotes $\mathbb{R}^{2\times d}$-to-$\mathbb{R}$ and $\mathbb{R}^{2\times d}$-to-$\mathbb{R}^d$ fully-connected layer, respectively, σ denotes the *sigmoid* function, and $diag(\cdot)$ converts the vector into a diagonal matrix. Besides, $Identity$ and $Zero$ operations are also incorporated in the candidate operation set to control the information flow.

Neighbor aggregation. Neighbor aggregation is a fundamental operation in the message-passing mechanism as it determines how nodes aggregate information from their neighbors, described as:

$$h_v^{(k)} = AGGREGATE(\{h_u^{(k-1)} : u \in N(v)\}), \tag{5}$$

where $h_v^{(k)}$ denotes the feature vector of node v at the k-th iteration of aggregation, $N(v)$ refers to the neighbors of node v, and $AGGREGATE(\cdot)$ denotes the neighbor aggregation operation. Different neighbor aggregation operations may excel at capturing different types of information. For instance, the *sum* aggregator is effective in capturing the overall information of neighboring nodes, while the *max* aggregator can extract the most prominent information among the neighboring nodes, and the *mean* aggregator captures statistics from the received message, which can help to mitigate the effects of extreme values. Since their different properties and advantages, multiple aggregators can be combined to exploit the potential of GNN in capturing spatial correlations among different time series.

Adaptive adjacency matrix. The GNN approaches rely heavily on graph structure for information propagation. However, in most cases, multivariate time series has no off-the-shelf graph structure since the relationships between time series are not known in advance. Manually constructing a graph structure requires extensive prior knowledge of the relevant task and may not accurately reflect the genuine connection between time series. For this purpose, we employ a graph learning layer to learn an adaptive adjacency matrix from data, which can be formulated as:

$$M_1 = tanh(\alpha E_1 \Theta_1),\ M_2 = tanh(\alpha E_2 \Theta_2), \tag{6}$$

$$A = ReLU(tanh(\alpha(M_1 M_2^T - M_2 M_1^T))), \tag{7}$$

where E_1 and E_2 denote the randomly initialized node embeddings, $\Theta 1$ and $\Theta 2$ denote the learnable weight matrixs, α is the scaling factor of the activation function. To reduce the computation cost of spatial convolution, we retain the neighbors with the top-k largest value in the adjacency matrix for each node, and set the value of the reset other nodes in the adjacency matrix to zero. The adaptive adjacency matrix is learned in an end-to-end manner through gradient descent.

3.2. Temporal Convolution Module

The temporal convolution module employs dilated causal convolution to construct TCN for capturing temporal dependencies within the time series. The dilated causal convolution sequentially slides over past observations by skipping values with a certain step to forecast the future value, which is an efficient and powerful tool for modeling sequential data. Mathematically, given a 1D sequence input $\mathbf{z} \in \mathbb{R}^T$, the dilated causal convolution operation of $\mathbf{z}$ at time step t can be formulated as:

$$\mathbf{z} \star f_{1\times k}(t) = \sum_{s=0}^{k-1} f_{1\times k}(s)\mathbf{z}(t - d \times s), \tag{8}$$

where $f_{1\times k} \in \mathbb{R}^k$ denotes the 1D convolution filter with kernel size $1 \times k$, and d denotes the dilation factor. By stacking dilated causal convolution filters, the receptive field of

the TCN grows exponentially, which enables its capability to handle longer sequences. However, simply stacking convolution filters with identical filter sizes may not be sufficient to effectively capture both short-term and long-term temporal patterns. To address this problem, we incorporate dilated causal convolution with different filter sizes in our search space, and explore their combination to come up with the most optimal TCN for capturing the temporal dependencies inherent in the given dataset.

Figure 1(c) illustrates the search space of the TCN, which is represented as a DAG consisting of an input node, an ordered sequence of N_t intermediate nodes, and an output node. Each intermediate node represents a latent representation, and has an edge associated with an operation that takes input from the input node or the previous node. The output node collects the latent representations of all the intermediate nodes and concatenates them across the channel dimension to form the output of the TCN, described as :

$$\xi' = Concat(\mathbf{z} \star o_1, \mathbf{z} \star o_2, \ldots, \mathbf{z} \star o_{N_t}), \tag{9}$$

where o_1, o_2 and o_{N_t} denote the operation from the candidate operation set. After that, the gating mechanism is employed to control the information flow to the next module. To be specific, we feed the output of two TCNs to two different activation functions to get the final output of the temporal convolution module, respectively. This computation procedure can be formulated as:

$$\xi = Tanh(\xi'_1) \odot Sigmoid(\xi'_2), \tag{10}$$

where $Tanh(\cdot)$ denotes the tangent hyperbolic activation function, and $Sigmoid(\cdot)$ denotes the sigmoid activation function. Following Wu et al. (2020), we incorporate 1×2, 1×3, 1×6, and 1×7 dilated casual convolution filters into the candidate operation set, the *Identity* and *Zero* operations are also included as the convention in NAS. By combining dilated causal convolution with different filter sizes, our temporal convolution module can effectively capture the complex temporal dependence under various ranges, enabling the tailored neural architecture design for the target task.

3.3. Evolutionary Search Algorithm

Evolutionary neural architecture search approaches solve the problem of automated neural architecture design by utilizing evolutionary computation techniques Davis (1991), which are a class of population-based paradigms that simulate the process of species evolution or population behavior in nature. In EMTSF, we propose a genetic algorithms based evolutionary search algorithm to heuristically discover the best spatial and temporal convolution modules for multivariate time series forecasting.

Algorithm 1 shows the framework of the proposed evolutionary search algorithm. Specifically, we start by initializing the population, each individual is randomly generated with the proposed gene encoding strategy. Then, the fitness evaluation proceeds, using conventional training to get the validation accuracy of each individual on the target dataset as its fitness value. After that, parent individuals are selected based on their fitness value with the binary tournament selection Miller et al. (1995), and then generate their offspring with the proposed genetic operations (i.e., crossover and mutation). Next, the fitness value of the offspring individuals is evaluated, and the parent and offspring populations will undergo an

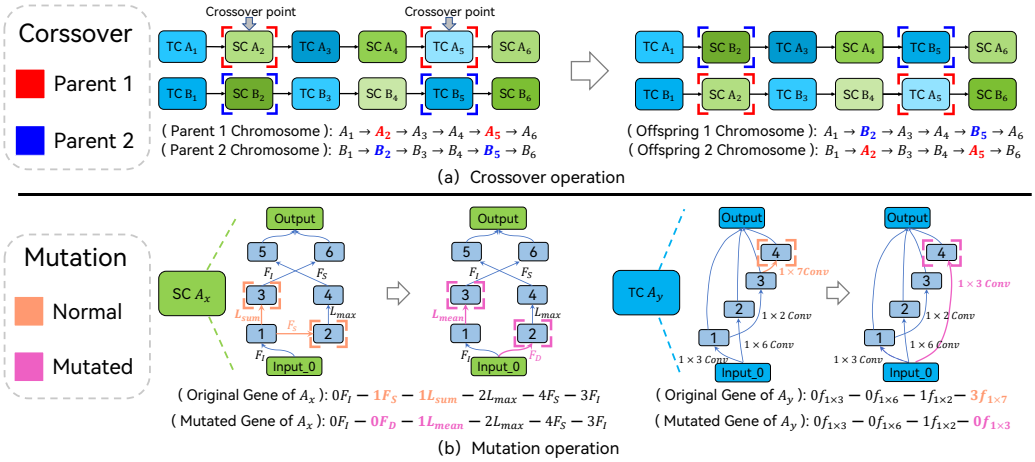


Figure 2: An illustration of the proposed genetic operations. The crossover operation (a) randomly selects several crossover points at the parent chromosomes and swaps the selected spatial/temporal convolution modules to generate new offspring. The mutation operation (b) modifies the interior of the convolution module by randomly selecting several pairs of indicators and replacing them with random new indicators to create new offspring. SC and TC represent the spatial convolution module and the temporal convolution module, respectively. F_S, F_D, F_I are feature filtering operations and L_{sum}, L_{max}, L_{mean} are neighbor aggregation operations.

Algorithm 1: Framework of the evolutionary search algorithm

$P_0 \leftarrow$ Initialize a population with the given population size using the proposed encoding stragety;
Evaluate the fitness of each individual in P_0;
$t \leftarrow 0$;
while $t <$ *the maximal generation number* **do**
  $Q_t \longleftarrow$ Generate offspring from the selected parent individuals using the proposed mutation and the crossover operators;
  Evaluate the fitness of each individual in Q_t;
  $P_{t+1} \leftarrow$ Environment selection from $P_t \cap Q_t$;
  $t \leftarrow t + 1$;
end
Return the individual having the best fitness in P_t.

environmental selection process to determine the individuals that will survive to the next generation. The evolution continues the until the predefined stopping criterion is satisfied.

Encoding strategy. The encoding strategy encodes neural architectures as a set of chromosomes that can be manipulated through genetic operators. In EMTSF, we encode the computation cells of spatial and temporal convolution modules as a set of strings (gene). Specifically, each intermediate node in the computation cell is represented by two identifiers,

i.e., I_i and O_i. The former identifier I_i specifies where the current node receives its input from, while the latter identifier O_i specifies the corresponding operation performed on that input. According to the order of the intermediate nodes, these identifiers will be combined to form a string that describes the entire module (see Figure 2(b)).

Genetic operations. Genetic operations (i.e., crossover and mutation) are utilized to create new offspring by modifying the parent individuals. The crossover operation modifies the chromosomes of individuals at the gene level. It involves randomly selecting several crossover points on the parent chromosome and exchanging the pointed genes (i.e., swapping the spatial/temporal convolution modules at selected positions between two architectures) to create two new offspring, as illustrated in Figure 2(a). On the other hand, the mutation operation introduces modifications within the gene. It involves selecting several pair of indicators inside the gene and replacing them with new random pair of indicators (i.e., to modify the computation cell of the selected spatial/temporal convolution modules) to generate a new offspring, as shown in Figure 2(b). These two operations together enable the exploration and exploitation of a diverse range of neural architectures, leading to the discovery of promising solutions.

Environment selection. The goal of environment selection is to maintain a diverse population while encouraging promising convergence. To achieve this, we employ two selection strategies: elitism Bhandari et al. (1996) and binary tournament selection. Specifically, we first explicitly add the top N_{elite} individuals to the next population. After that, we conduct binary tournament selection to select individuals from the union of the current population and offspring, adding them to the next population until the predefined population size is reached.

4. Experiments

In this section, we conduct a series of experiments to demonstrate the effectiveness of EMTSF in automating neural architecture design for multivariate time series forecasting.

4.1. Datasets

We evaluate EMTSF on two types of time series forecasting tasks using four real-world datasets across various domains, the brief statistical information can be seen in Table 1. For multi-step forecasting tasks, we employ the widely used METR-LA and PEMS-BAY datasets Li et al. (2018), which are split chronologically into 70% for training, 10% for validation, and 20% for testing. Similarly, for single-step forecasting tasks, we use the Electricity and Exchange-rate datasets Lai et al. (2018), which are split chronologically into 60% for training, 20% for validation, and 20% for testing.

Table 1: Dataset statistics.

Datasets	Samples	Nodes	Sample Rate	Input Length	Output Length
METR-LA	34272	207	5 minutes	12	12
PEMS-BAY	52116	325	5 minutes	12	12
Exchange-Rate	7588	8	1 day	168	1
Electricity	26304	321	1 hour	168	1

Following previous studies Wu et al. (2020), our evaluation metrics for the multi-step forecasting task include Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). As for single-step forecasting, the evaluation metrics consist of Root Relative Squared Error (RRSE) and Empirical Correlation Coefficient (CORR). Lower values of RMSE, MAE, MAPE, and RRSE indicate better performance, while higher values of CORR are desirable.

4.2. Baseline

We compare EMTSF with two groups of baseline methods for multi-step and single-step forecasting, respectively.

Multi-step forecasting. We have chosen seven competitive spatial-temporal methods, namely DCRNN Li et al. (2018), STGCN Yu et al. (2017), ST-MetaNet Pan et al. (2019), AGCRN Bai et al. (2020), GMAN Zheng et al. (2020), MTGNN Wu et al. (2020), and AutoSTG Pan et al. (2021). Among them, all these methods, except AutoSTG, are manually designed by human experts, while AutoSTG employs an automated approach.

Single-step forecasting. We have selected eight popular time-series forecasting methods, which encompass classical statistical methods such as Auto-Regressive (AR) and Gaussian Process (GP) Roberts et al. (2013), as well as deep learning methods such as RNN-GRU, VARMLP Zhang (2003), LSTNet Lai et al. (2018), TPA-LSTM Shih et al. (2019), MTGNN Wu et al. (2020), and StemGNN Cao et al. (2020).

4.3. Experimental Setups

EMTSF is implemented by Pytorch 2.0 and DGL 1.0, and all the experiments are conducted on an Ubuntu server equipped with four Nvidia GeForce RTX 2080Ti GPUs. In the following test, we set both the number of spatial and temporal convolution modules to 3, and the dilation factor of each temporal convolution module is set to 1, 2, and 4, respectively. The spatial and temporal convolution modules have 6 and 4 intermediate nodes, respectively. The model is trained using the Adam optimizer with a learning rate of 0.001 and gradient clipping with a threshold of 5. The settings of the graph learning layer are identical to those in its original paper Wu et al. (2020). For evolutionary search, the population size and the number of generations are all set to 20, and the probabilities of crossover and mutation are set to 0.9 and 0.5, respectively. In the environment selection, the number of elites is specified as 4 (i.e., 20% of the population size). After the evolutionary search, we retrain the architecture with the highest fitness 10 times to report its final performance.

4.4. Main Results

Tables 2 and 3 present the performance comparison between EMTSF and the baseline methods for multi-step and single-step forecasting, respectively. The best result is highlighted in bold font, and the second-best result is underlined. It is evident that EMTSF outperforms most of the baseline methods, demonstrating state-of-the-art performance in both tasks. In the following sections, we will individually discuss the results of multi-step and single-step forecasting.

Multi-step forecasting. In this task, we compare EMTSF with several spatial-temporal methods, most of which are mainly designed for the traffic prediction domain.

Table 2 shows the performance comparison of baseline models and EMTSF on multi-step forecasting, where the evaluation metrics MAE, RMSE, and MAPE at the next 3/6/12 timestamps are presented. In general, our EMTSF achieves comparable performance to state-of-the-art spatial-temporal methods without leveraging any prior knowledge. On the contrary, DCRNN, STGCN, and GMAN rely on pre-defined graphs with additional domain knowledge (e.g., leverage road distance, GPS locations, and nearby points of interest). While methods like AGCRN and MTGNN can eliminate the reliance on domain knowledge, their architectures are manually designed by human experts, which are difficult to adjust themselves corresponding to the data. AutoSTG employs NAS to generate data-specifical architectures, but it still relies on additional domain knowledge from the traffic domain to enhance its forecasting accuracy. Furthermore, it adopts pre-defined and fixed components as the candidate operations without any inside search, limiting its ability to achieve more accurate forecasting.

Table 2: Performance comparison of baseline models and EMTSF on multi-step forecasting.

Data	Models	Horizon 3			Horizon 6			Horizon 12		
		MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
METR-LA	DCRNN	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
	STGCN	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
	ST-MetaNet	2.69	5.17	6.91%	3.10	6.28	8.57%	3.59	7.52	10.63%
	AGCRN	2.85	5.47	7.60%	3.22	6.56	8.83%	3.61	7.46	10.23%
	GMAN	2.77	5.48	7.25%	3.07	6.34	8.35%	**3.40**	**7.21**	**9.72%**
	MTGNN	2.69	5.18	6.86%	3.05	6.17	8.19%	3.49	7.23	9.87%
	AutoSTG	2.70	5.17	6.93%	3.07	6.19	8.37%	3.47	7.29	9.85%
	EMTSF	**2.67**	**5.15**	**6.80%**	**3.04**	**6.15**	**8.13%**	3.47	**7.21**	9.94%
PEMS-BAY	DCRNN	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
	STGCN	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
	ST-MetaNet	1.36	2.90	2.82%	1.76	4.02	4.00%	2.20	5.06	5.45%
	AGCRN	1.35	2.83	2.87%	1.69	3.81	3.84%	1.96	4.52	4.67%
	GMAN	1.34	2.82	2.81%	**1.62**	3.72	3.63%	**1.86**	4.32	**4.31**%
	MTGNN	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	4.53%
	AutoSTG	1.33	2.79	2.78%	1.64	3.68	3.65%	1.92	4.53	4.45%
	EMTSF	**1.31**	**2.77**	**2.73%**	**1.62**	**3.65**	**3.60%**	1.88	**4.31**	4.41%

Single-step forecasting. In this task, we compared EMTSF with several multivariate time series methods. The detailed experiment results for single-step forecasting are presented in Table 3, which includes the evaluation metrics RRSE and CORR for each method at the next 3/6/12/24 timestamps. It is evident that the statistical methods AR and GP perform poorly compared to the deep-learning-based methods, which can be attributed to their reliance on stationary assumptions of time series. In contrast, deep-learning-based methods are free from such assumptions and excel in capturing complex non-linear relationships among multivariate time series. However, the hand-crafted architectures of deep-learning-based methods limit their ability to adapt to different datasets. This is supported by the fact that no single model can be optimal on both datasets. For instance, TPA-LSTM, which performs the second best on the exchange-rate dataset, does not perform as well as MTGNN on the electricity dataset. In contrast, the proposed EMTSF can automate the design of neural architectures for different datasets, allowing it to achieve state-of-the-art performance on both datasets.

Table 3: Performance comparison of baseline models and EMTSF on single-step forecasting.

Dataset		Exchange-Rate				Electricity			
		Horizon				Horizon			
Methods	Metrics	3	6	12	24	3	6	12	24
AR	RSE	0.0228	0.0279	0.0353	0.0445	0.0995	0.1035	0.1050	0.1054
	CORR	0.9734	0.9656	0.9526	0.9357	0.8845	0.8632	0.8591	0.8595
GP	RSE	0.0239	0.0272	0.0394	0.0580	0.1500	0.1907	0.1621	0.1273
	CORR	0.8713	0.8193	0.8484	0.8278	0.8670	0.8334	0.8394	0.8818
VARMLP	RSE	0.0265	0.0394	0.0407	0.0578	0.1393	0.1620	0.1557	0.1274
	CORR	0.8609	0.8725	0.8280	0.7675	0.8708	0.8389	0.8192	0.8679
RNN-GRU	RSE	0.0192	0.0264	0.0408	0.0626	0.1102	0.1144	0.1183	0.1295
	CORR	0.9786	0.9712	0.9531	0.9223	0.8597	0.8623	0.8472	0.8651
LSTNet	RSE	0.0226	0.0280	0.0356	0.0449	0.0864	0.0931	0.1007	0.1007
	CORR	0.9735	0.9658	0.9511	0.9354	0.9283	0.9135	0.9077	0.9119
TPA-LSTM	RSE	0.0174	0.0241	0.0341	0.0444	0.0823	0.0916	0.0964	0.1006
	CORR	0.9790	0.9709	0.9564	0.9381	0.9439	0.9337	0.9250	0.9133
MTGNN	RSE	0.0194	0.0259	0.0349	0.0456	0.0745	0.0878	0.0916	0.0953
	CORR	0.9786	0.9708	0.9551	0.9372	0.9474	0.9316	0.9278	0.9234
StemGNN	RSE	0.0506	0.0674	0.0676	0.0685	0.0799	0.0909	0.0989	0.1019
	CORR	0.8871	0.8703	0.8499	0.8738	**0.9490**	**0.9397**	**0.9342**	0.9209
EMTSF	RSE	**0.0173**	**0.0238**	**0.0339**	**0.0441**	**0.0740**	**0.0872**	**0.0908**	**0.0947**
	CORR	**0.9792**	**0.9714**	**0.9567**	**0.9386**	0.9482	0.9330	0.9291	**0.9250**

4.5. Ablation Study

In this subsection, we conduct an ablation study on the METR-LA dataset to verify the effectiveness of the key components in EMTSF. The compared variants of EMTSF are introduced as follows:

• **w/o SC:** In this variant, we remove the spatial convolution module in EMTSF and use a fully connected layer to replace it instead.

• **w/o TC:** In this variant, we remove the temporal convolution module and use a single 1x7 dilated convolution filter with the same output channel to replace it.

• **Random:** In this variant, we randomly sample architectures from our search space of spatial and temporal convolution modules and train them directly for prediction.

• **w/o crossover:** In this variant, we perform the evolutionary neural architecture search without crossover operation when generating new offspring.

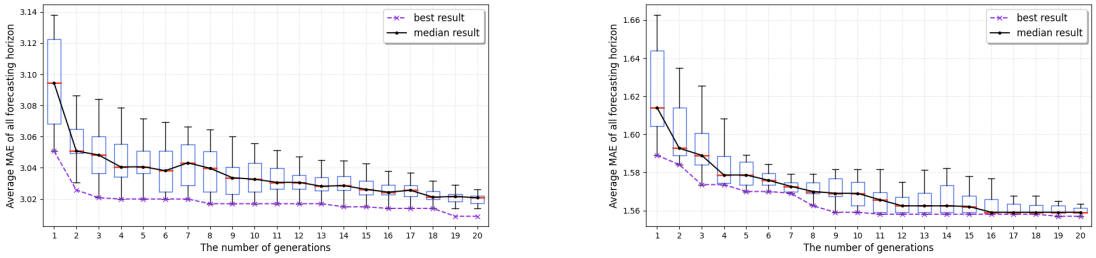
• **w/o mutation:** In this variant, we perform the evolutionary neural architecture search without mutation operation when generating new offspring.

Table 4 shows the performance of EMTSF and the compared variants on the METR-LA dataset, where the average MAE, MAPE, and RMSE of all forecasting horizons (i.e., the future 12 timesteps) are presented. As can be observed, both the spatial and temporal convolution modules contribute to the performance of EMTSF to a certain extent. When replacing the spatial convolution module with a fully connected layer, dramatic degradation of model performance occurs since it can not effectively capture the spatial correlation among time series. The replacement of the temporal convolution module also affects the model performance, but since the 1x7 dilated causal convolution filter still holds the capability to model temporal dependence, the model performance does not show significant degradation. In terms of architecture search, the comparison between EMTSF and the Random variant demonstrates the superiority of the proposed evolutionary search algorithm in

identifying promising architectures. The result of w/o crossover and w/o mutation variants demonstrates the importance of the genetic operation in exploring the search space. Using only the crossover operation leads to a lack of diversity in the population, limiting the exploration ability. On the other hand, using only mutation operation restrict the exploitation of parent generation, compromising the stability of the search process.

Table 4: Ablation Study.

Method	MAE	RMSE	MAPE
w/o SC	3.1304±0.0084	6.3802±0.0296	0.0875±0.0010
w/o TC	3.0165±0.0063	6.0583±0.0274	0.0816±0.0009
Random	3.0812±0.0366	6.3103±0.0878	0.0857±0.0024
w/o crossover	3.0174±0.0324	6.0458±0.0762	0.0821±0.0022
w/o mutation	3.0536±0.0075	6.2989±0.0328	0.0849±0.0012
EMTSF	**3.0089±0.0053**	**6.0372±0.0244**	**0.0815±0.0007**



(a) METR-LA (b) PEMS-BAY

Figure 3: Evolutionary trajectory of the proposed EMTSF in discovering the best architecture on the (a) METR-LA and (b) PEMS-BAY dataset.

4.6. Evolutionary Trajectories

To better analyze the convergence of the evolutionary search algorithm in identifying promising architectures for multivariate time series forecasting, we collect the individuals selected through environmental selection in each generation and present their statistical results using box plots. Figure 3 showcases the evolutionary trajectories of EMTSF on the METR-LA and PEMS-BAY datasets. The horizontal axis represents the number of generations, while the vertical axis denotes the overall MAE of the individuals (i.e., the average MAE across all forecasting horizons). We connect the best and median overall MAE in each generation with a violet dashed line and a black solid line, respectively. As can be seen, the median MAE and minimum MAE exhibit a decreasing trend as the evolution progresses. During the first two generations, there is a notable decline in the MAE for both datasets, which can be attributed to the random initialization of the population at the start of the evolution. In the subsequent generations, the minimum MAE steadily decreases, which is achieved through the utilized elitism mechanism in the environment selection. Meanwhile, the rate of decline in the median MAE and minimum MAE gradually decreases, accompanied by

a decreasing trend in the height of the box plot, which indicates that the search is moving towards a more stable state. In conclusion, the proposed search algorithm converges within the default parameter settings, enabling users to utilize EMTSF effectively in finding optimal neural architectures for their own data.

5. Conclusion

In this paper, we proposed EMTSF, a novel NAS framework that automates neural architecture design for multivariate time series forecasting. Our framework incorporates a fine-grained search space for both the spatial and temporal modules, enabling us to discover powerful message-passing mechanisms and optimal combinations of temporal convolutions. Besides, we employ an evolutionary search algorithm to exploit the search space and effectively design promising neural architecture for the given data without any expert knowledge. Extensive experiments conducted on four real-world benchmark datasets demonstrate the superiority of the proposed EMTSF over the baselines. This research opens new possibilities for users interested in multivariate time series forecasting, providing an accessible solution that eliminates the need for expert knowledge and manual labor.

References

James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33:17804–17815, 2020.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Dinabandhu Bhandari, CA Murthy, and Sankar K Pal. Genetic algorithm with elitist model and its convergence. *International Journal of Pattern Recognition and Artificial Intelligence*, 10(06): 731–747, 1996.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6657–6666, 2021.

Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in Neural Information Processing Systems*, 33:17766–17778, 2020.

Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. *Advances in Neural Information Processing Systems*, 32, 2019.

Lawrence Davis. Handbook of genetic algorithms. 1991.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Roger Frigola. *Bayesian time series learning with Gaussian processes*. PhD thesis, University of Cambridge, 2015.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Bo Jin, Haoyu Yang, Leilei Sun, Chuanren Liu, Yue Qu, and Jianing Tong. A treatment engine by predicting next-period prescriptions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1608–1616, 2018.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR '18)*, 2018.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 82–92, 2019a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *International Conference on Learning Representations*, 2019b.

Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1720–1730, 2019.

Zheyi Pan, Songyu Ke, Xiaodu Yang, Yuxuan Liang, Yong Yu, Junbo Zhang, and Yu Zheng. Autostg: Neural architecture search for predictions of spatio-temporal graph. In *Proceedings of the Web Conference 2021*, pages 1846–1855, 2021.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.

Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108:1421–1441, 2019.

Jelena Simeunović, Baptiste Schubnel, Pierre-Jean Alet, and Rafael E Carrillo. Spatio-temporal graph neural networks for multi-site pv power forecasting. *IEEE Transactions on Sustainable Energy*, 13(2):1210–1220, 2021.

Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Jiancheng Lv. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics*, 50(9):3840–3854, 2020.

Chakkrit Termritthikun, Yeshi Jamtsho, Jirarat Ieamsaard, Paisarn Muneesawang, and Ivan Lee. Eeea-net: An early exit evolutionary neural architecture search. *Engineering Applications of Artificial Intelligence*, 104:104397, 2021.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*, 2019.

Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2296–2306, 2022.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2141–2149, 2017.

Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1234–1241, 2020.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.