An evolutionary neural architecture search-based approach for time series forecasting.

VU, T.H., NGUYEN, T.T. and ELYAN, E.

2025

An Evolutionary Neural Architecture Search-Based Approach for Time Series Forecasting

1st Trung Hieu Vu
School of Computing, Engineering and Technology, Robert Gordon University
Aberdeen, UK
h.vu1@rgu.ac.uk

2nd Tien Thanh Nguyen
School of Computing, Engineering and Technology, Robert Gordon University
Aberdeen, UK
t.nguyen11@rgu.ac.uk

3rd Eyad Elyan
School of Computing, Engineering and Technology, Robert Gordon University
Aberdeen, UK
e.elyan@rgu.ac.uk

***Abstract*—Time series forecasting (TSF) is one of the most prevalent research topics in artificial intelligence and has garnered significant attention in the research community. In recent years, significant breakthroughs have been made in TSF research, shifting from traditional statistical models to deep learning (DL), and attention-based methods. While attention-based methods excel at capturing global dependencies, they often face challenges in effectively modeling local patterns. Additionally, the design of these networks typically demands substantial human expertise, experimental work, and manual configuration. To address these issues, we propose an Evolutionary Neural Architecture Search for Time Series Forecasting, entitled ENAS-TSF to automate TSF architecture design. Concretely, we propose a novel local/ global context module encoding strategy to define a search space. Each local context module includes various convolutions with different kernel sizes to capture temporal dependencies, aiming to enhance the local features and pattern recognition. Global encoding meanwhile contains attention mechanisms and feed-forward layers for global context modeling. We propose an evolutionary neural architecture search approach to identify the optimal ENAS-TSF architectures, achieving the ideal balance of local/ global context modeling. Extensive experiments on common benchmark datasets show that ENAS-TSF achieves competitive performance compared to state-of-the-art methods, demonstrating the proposed framework's effectiveness.**

***Index Terms*—Time series forecasting, neural architecture search, deep learning, evolutionary algorithm.**

I. Introduction

Time series data refers to a sequence of observations ordered by discrete time, which is captured from various sensors in practice with multiple variables. The time series data plays a crucial role in real-world applications, such as weather forecasting, intelligent transportation systems, and energy consumption. With the increasing development of advanced technologies such as the Internet of Things (IoT), smart devices, and edge computing, massive amounts of time series data are generated in real-time across various domains. This poses many challenges for the time-series analysis community. Compared with computer vision (CV) [1] and natural language processing (NLP) [4], which are relatively intuitive patterns, it is required to discover intricate temporal patterns and the complex dependencies among different sequence variables of time series data.

Significant breakthroughs have been witnessed in the time series community in recent years, shifting from traditional statistical time series models to more sophisticated deep learning techniques. Several prominent statistical time series models such as Auto Regressive Integrated Moving Average (ARIMA) [2], Exponential Smoothing (ETS) [3], and Seasonal-Trend Decomposition (STD) [3] offer as forecaster's toolboxes, which are useful for many forecasting situations. These methods rely on statistics and mathematical equations to uncover characteristics of time series patterns, including trends, seasonality, and cycles, or find relationships between past observations and predictions for the future. However, statistical methods are bound by assumptions such as stationery and linearity, which limit the potential of this approach in practice. Deep Learning approaches have emerged as a powerful solution to address the limitations and achieve outstanding performance in the field of time series data analysis. Various deep learning architectures originating from the NLP and CV communities [1], [4] have been proposed and adapted to fit time series data. These approaches have demonstrated the ability to capture long-term dependencies and intricate non-linear relationships in real-world applications, yielding promising results. Some notable works include TimesNet [6], FEDformer [7], Autoformer [5], Crossformer [8], and Informer [9].

In this paper, we aim to integrate the strengths of CNN-based methods and self-attention mechanisms to model better the inherent characteristics of time series sequences, such as local temporal features and global correlations. The local temporal features refer to the properties of a temporal sequence across a small period τ, and global correlations represent the complex interactions and associations among many periods $\{\tau_1, \tau_2, \ldots, \tau_n\}$. For instance, energy consumption is not only affected by daily usage habits but may also be correlated with larger trends over time, such as monthly patterns or changing seasons. By exploring the underlying characteristics of a given time period and the relationships among previous periods, we can identify a more accurate estimation of the value at a specific time point. Therefore, a robust forecasting model should exhibit two essential properties: (1) the ability to extract

local features that effectively capture short-term variations and (2) the ability to model global correlations that illustrate long-term relationships. While models like Conformer [12], and MICN [16] have been proposed to combine local and global modeling for greater efficiency, they were manually designed through a trial-and-error process, making them time-consuming.

To address the above problems, we proposed an Evolutionary Neural Architecture Search for Time Series Forecasting (**ENAS-TSF**). We defined a search space on multiple branches in which for each branch, we quipped a local module with different convolutions to model its local context, modeling information of the sequence separately. Moreover, we modeled the global correlations using global modules to capture global correlations. Then, we proposed an Evolutionary Neural Architecture Search (ENAS) approach, which aims to automatically search for optimal local and global modules for striking the balancing of local/ global context modeling. Finally, the merge module is utilized to integrate information from various patterns across multiple branches. Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to apply NAS to search the DL architectures for the long-term sequence forecasting task. Here, our proposed ENAS-TSF method aims to identify the optimal modules and the ideal balancing of the local/ global context modeling. The proposed method explores optimal architecture solutions for different time series datasets and achieves strong prediction performance for the long-term sequence forecasting task.
- In the proposed ENAS-TSF, we designed a novel search space for multi-branch DL architectures and proposed an effective ENAS to explore the search space. The search space encoding includes the local module to extract the better local temporal feature and the global model to capture global correlations. Our proposed ENAS-TSF method employs an early-generation stopping strategy to accelerate training networks and convergence.
- We conducted extensive experiments on benchmark real-world time series datasets to verify the proposed ENAS-TSF approach. The experimental results demonstrate that the architectures found by our ENAS-TSF method show promising performance compared to state-of-the-art TSF baselines.

II. Related Work

As discussed above, ENAS aims to search optimal architectures of DNNs for time series forecasting tasks. In this section, we will mention the time series forecasting models in Section II-A and a brief review of the evolutionary neural architecture search in Section II-B.

A. Time Series Forecasting Models

Time series forecasting approaches have evolved significantly, transitioning from traditional statistical models to complicated DL models. Many statistical models remain widely used and are adopted as foundational design principles incorporating advanced DL architectures. In the early stage, prominent statistical approaches such as ARIMA [2], ETS [3], auto-regressive model (VAR) [3], etc. can be mentioned. Besides, recent years have witnessed the development of advanced DL methods, and these methods have been proposed for time series forecasting. These methods can be classified into five categories based on backbone architectures: RNN-based, CNN-based, MLP-based, GNN-based, and Transformer-based approaches [13]. The RNN-based time series models [14], [15] utilize the recurrent structure to capture temporal dependencies by hidden state transformation among time steps and modeling the mutual correlation among multivariate variables. CNN-based models [16], [17] become competitive time series backbones with capturing better local features and pattern recognition. MLP-based models [11] inspired by auto-regressive models become a popular approach for modeling temporal data. The GNN-based architectures [18], aiming to capture underlying topological relations in multivariate time series data, serve as a spatio-temporal graph. Finally, Transformer-based models nowadays have emerged as a powerful solution for analyzing time series data based on advantages from attention mechanisms. The Transformer-based models [5], [19] can capture long-term temporal dependencies and multivariate correlations.

B. Evolutionary Neural Architecture Search

The ENAS algorithms address the NAS problem based on using evolutionary computation (EC) techniques. EC refers to a group of population-based computational approaches that mimic the process of natural evolution or population behaviors in nature to tackle complex optimization problems. Specifically, genetic algorithms (GAs), genetic programming (GP), and particle swarm optimization (PSO) are commonly utilized EC techniques. The first work of ENAS, namely the LargeEvo algorithm [20], was proposed by the Google Brain team. The proposed method used a GA algorithm to search for the best CNN architectures on CIFAR-10 and CIFAR-100 datasets. Xie and Yuille [21] introduced the Genetic CNN, where the authors encoded the network structure using fixed-length binary strings. Then, different genetic operations such as selection, mutation, and crossover are applied to find high-quality candidate solutions. However, a fixed-length binary string encoding can limit the length of deep structures. Sun et al. [22] proposed a variable-length gene encoding strategy, allowing the generation of deeper architecture configurations. However, this flexible approach can lead to excessively long architectures, resulting in excessive model size and slow training speed.

Although many works utilize NAS approaches to optimize domains such as NLP and CV, these design search spaces for different domains cannot be directly applied to time series forecasting and the problem of balancing local context and global context in this paper. To the best of our knowledge, we are the first to apply the ENAS to search DL architectures for the long-term time series forecasting task. It is noted that our

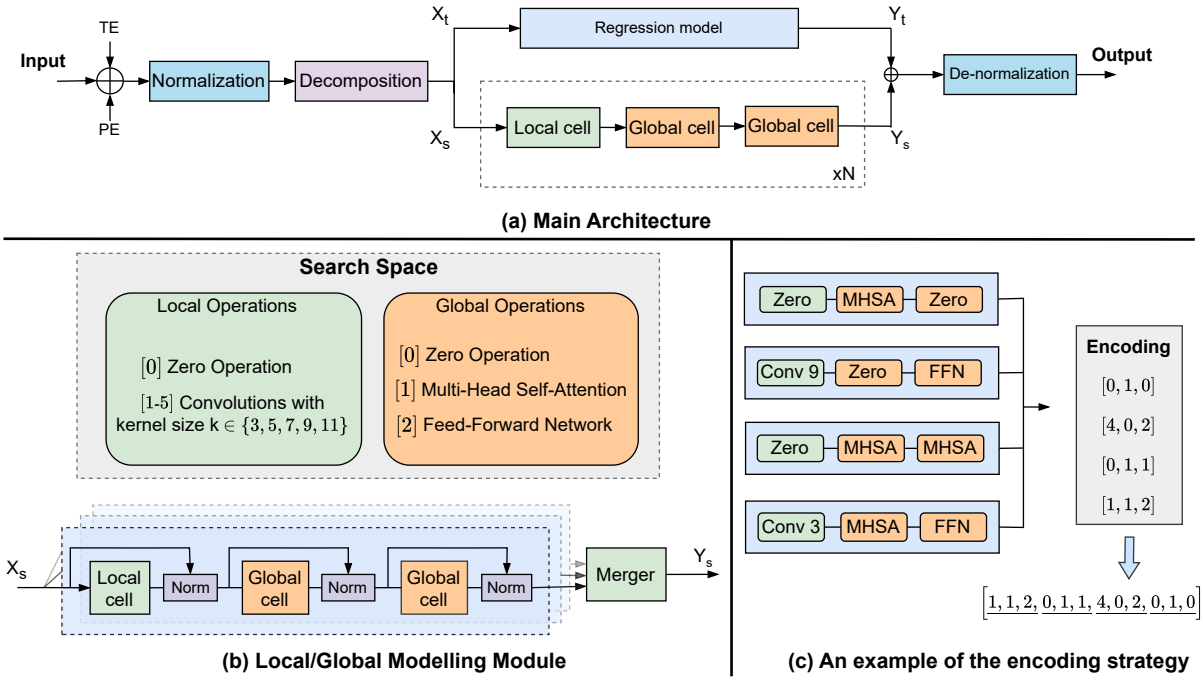


Fig. 1: The overall ENAS-TSF architecture. Fig. 1(a) presents the main architecture of ENAS-TSF. Fig. 1(b) provides the detailed structure and search space of the local/ global modeling module. Fig. 1(c) shows an illustrative example of the utilized encoding strategy.

work is different from the work of Liang et al. [23], who proposed an evolutionary neural architecture search framework, entitled EMTSF, for the automated design of spatial-temporal graph neural networks (STGNNs). Their proposed method aims to optimize the GNN architectures for temporal and spatial time series datasets, which is distinct from long-term time series forecasting.

III. Methodology

We propose an Evolutionary Neural Architecture Search for Time Series Forecasting, called ENAS-TSF, to automate the design of deep architectures for handling the balancing local/ global modeling patterns. The objective of long-term time series prediction is to predict a future sequence of length O using a historical sequence of length I. The relationship can be described as input I - output O, where the output O is much substantially greater than the input I. In this section, we will present the detailed workflow of the entire framework and explain how we use an evolutionary algorithm to search for optimal deep architectures.

A. The proposed ENAS-TSF

The main architecture of the proposed framework can be referred to in Fig. 1(a). The original input data is initially projected into an embedding input. Then, the positional information, including temporal encoding (TE) and positional encoding (PE), is injected into the embedding input to help the model utilize the order of the time series sequence effectively. To overcome the non-stationary of data, which is the inherent property in real-world time series, we adopt the series stationarization, including the Normalization module and De-normalization module from Non-stationary Transformers [24]. The Normalization module serves as a pre-processing step to deal with non-stationary series caused by varied mean and standard deviation. The De-normalization module works as a post-processing at the end to transform the model outputs back with original statistics. Inspired by decomposition techniques in time series analysis [5], we adopt a decomposition block to separate complex pattern time series data into season patterns and trend-cyclical. The season part modeling is the input of the backbone of the proposed architecture, which includes local cell modules and global cell modules from different branches. This design aims to automatically designed to capture more accurate local features and global correlations. Additionally, the skip connections are incorporated with the local/ global modeling module to mitigate the vanishing/exploding gradient problem, as shown in Fig. 1(b). The trend-cyclical series input X_t is fed into a linear regression modeling to make a prediction about the trend-cyclical Y_t, which gives the model a holistic view of the trend direction and achieves effective performance in time series forecasting [16]. Finally, the trend-cyclical Y_t and season parts Y_s are combined and go through the de-normalization block to obtain a final output prediction. In the subsequent subsections, we will briefly describe the search space for the local modeling module and the global modeling module. Furthermore, we will introduce the evolutionary search algorithm employed in ENAS-TSF to explore

the designed search space and explore promising architectures for the time series forecasting task.

B. The Local/ Global Modelling Module

Since original time series points cannot be directly captured well by the single self-attention mechanisms, we aim to incorporate the self-attention mechanism with the local context modeling. As shown in Fig. 1(b), the input X_s is passed through a local cell operation (LO) that focuses on capturing local feature context within data. Two global cell operations (GO) are designed to capture global correlations within data, which are inspired by the self-attention mechanism and feed-forward networks from Transformer architecture [4]. Among these cells, we applied layer normalization (LN) to ensure the data's stability before moving to the next stage [26]. Moreover, the residual path is utilized to avoid vanishing/exploding gradient among these stages. The local/global modeling module can be formalized as follows:

$$\begin{cases} lp = LN(X_s + LO(X_s)) \\ gp_1 = LN(lp + GO_1(lp)) \\ gp_2 = LN(gp_1 + GO_2(gp_1)) \end{cases} \tag{1}$$

where lp is the latent representation obtained by the local path, gp_1, gp_2 are the latent representation obtained by the global paths, $LO(\cdot)$, $GO_1(\cdot), GO_2(\cdot)$ denote a local operation, and global operations used for the local and global path, respectively.

The $LO(\cdot)$ offers six candidate operations, containing five one-dimensional convolution operations with different kernel sizes of $\{3, 5, 7, 9, 11\}$ and a zero operation that outputs a zero tensor. The zero operation is proposed to simplify the overall architecture. As demonstrated in Sandwich Transformer [25], reordering the multi-head self-attention (MHSA) and feed-forward network (FFN) can lead to better performance. This is the reason we use global cell operations GO to replace the order of the MHSA module and FFN module, which includes the MHSA operation, FFN operation, and a zero operation.

In this paper, the outputs from different branches in the local/global modeling module are merged using Conv2D with different weights instead of traditional Concat operation. These alternates yield a better performance in time series forecasting task [16].

C. Evolutionary Search Algorithm

1) Encoding strategy: We propose an encoding strategy in which each candidate architecture $\mathcal{A}$ is encoded into vector-based encoding:

$$\mathcal{A} = \big[(l_1, g_{11}, g_{12}), (l_2, g_{21}, g_{22}), \ldots, (l_N, g_{N1}, g_{N2})\big] \tag{2}$$

in which $\mathcal{A}$ includes N blocks (l_n, g_{n1}, g_{n2}) encoding for N layers, l_n denotes the local operators, $l_n \in LO = \{0, \cdots, 5\}$ including 6 predefined convolutions with different kernel sizes mentioned above, and (g_{n1}, g_{n2}) denote the global operators, $g_{n1}, g_{n2} \in GO = \{0, 1, 2\}$ including 3 predefined Zero operation, MHSA operation, and FFN operation for $n = \{1, \ldots, N\}$. By proposing the encoding in Equation 2, the search space size will be: $(|LO| * |GO| * |GO|)^N$, $|\cdot|$ is the cardinality of a set.

For a better understanding of the encoding strategy, Fig. 1(c) describes an illustrative example, where N layers are set to 4. The given architecture is encoded as follows: the first layer is represented by $[1, 1, 2]$, the second, third, and fourth layers are encoded as $[0, 1, 1]$, $[4, 0, 2]$, and $[0, 1, 0]$, respectively. As the result, the overall architecture's encoding $\mathcal{A}$ is $[(1, 1, 2), (0, 1, 1), (4, 0, 2), (0, 1, 0)]$. The architecture consists of four layers. Layer 1 is encoded as $[1, 1, 2]$, where 1 represents a convolution layer with a kernel size of 3, 1 indicates the MHSA cell, and 2 indicates the FFN. Based on the encoding of layer 2, we can see that only two MHSA cells are used. Layer 3 encoding, i.e. $[4, 0, 2]$ shows that this layer includes a convolution with a kernel size of 9, followed by a zero operation, and the FFN. Finally, Layer 4 is encoded as $[0, 1, 0]$, indicating that it includes only one MHSA cell.

Algorithm 1 Framework of the ENAS-TSF with Generation Convergence Early Stopping

Input: **D**: Training set, **V**: Validation set, Gen: Maximum number of generations, Pop: Population size, avg: the average fitness of population, $patience$: the maximum number of generations without improvement;
Output: P_b : The best individual;

$\mathcal{P} \leftarrow$ Randomly initialize Pop individuals;
Train each individual in $\mathcal{P}$ on **D** and evaluate its fitness on **V**;
$best\_fitness =$ Best fitness value from $\mathcal{P}$;
$no\_improvement = 0$
for $t = 0$ to Gen **do**
    $Q_t = \emptyset$;
    while $|Q_t| < Pop$ **do**
        Randomly select parents G_1, G_2 from $\mathcal{P}$;
        Generate offsprings P_1, P_2 from G_1, G_2 using Algorithm 2 and 3;
        Evaluate fitness of P_1, P_2 on **V**;
        $Q_t = Q_t \cup \{P_1, P_2\}$
    end while
    $\mathcal{P} =$ Select Pop best individuals from $\mathcal{P} \cup Q_t$;
    $current\_best =$ Best fitness value from $\mathcal{P}$;
    if $current\_best < best\_fitness$ **then**
        $best\_fitness = current\_best$;
        $no\_improvement = 0$;
    else
        $no\_improvement = no\_improvement + 1$;
    end if
    if $no\_improvement = patience$ **then**
        break;
    end if
end for
Select the best individual P_b from $\mathcal{P}$;
return: P_b;

Next, we search for the optimal DL architecture by solving the following optimization problem:

$$\min_{\mathcal{A}} \mathcal{F}(\mathcal{A}) = f_{\text{val\_MSE}}(\mathcal{A}, \mathbf{V}) \tag{3}$$

where $f_{\text{val\_MSE}}(\mathcal{A}, \mathbf{V})$ represents the Mean Square Error (MSE) fitness value of $\mathcal{A}$ on the validation set $\mathbf{V}$.

In this study, we use Evolutionary neural architecture search techniques to address the complex task of automating the design of neural architecture by leveraging evolutionary computation (EC) methods. These EC methods, rooted-in population-based approaches, draw inspiration from natural processes such as species evolution or population dynamic. In ENAS-TSF, we propose to use Genetic Algorithm, an evolutionary search algorithm to solve the optimization problem in (3).

2) Algorithm description: Algorithm 1 describes the proposed evolutionary search algorithm. Concretely, we start by initializing the population $\mathcal{P}$, and each individual is randomly generated with the proposed gene *encoding strategy*. Then, we train each individual on the training set $\mathbf{D}$ and then evaluate it on the validation set $\mathbf{V}$ to obtain its fitness value. Subsequently, an evolutionary process is carried out for a maximum number of generations Gen or until the stopping criterion is met. In each generation, pairs of parent candidates G_1, G_2 are selected randomly from $\mathcal{P}$, and offsprings P_1, P_2 are produced through the *genetic operations* such as crossover and mutation. These generated offsprings are then evaluated on the validation set $\mathbf{V}$, and the parent and offspring population undergo an environment selection process to determine the Pop best individuals for the next generation. To avoid unnecessary computations, we proposed a generation-convergence early stopping mechanism. The best fitness candidate in $\mathcal{P}$ is tracked across generations. If the best fitness value does not yield an improvement over a predefined $patience$ threshold, the search process terminates early, preventing redundant generations and reducing computational costs.

Algorithm 2 Single-Point Crossover

Input: G_1 and G_2: Two parents, p_c: Crossover probability, n: Length of parents;
Output: P_1 and P_2: Two offsprings;

r_c = Generate a random number between 0 and 1;
if $r_c < p_c$ **then**
  r_{pos} = Select a random position in the range $[1, n-1]$;
  Offspring $P_1 = G_1[0 : r_{pos}] + G_2[r_{pos} : n]$
  Offspring $P_2 = G_2[0 : r_{pos}] + G_1[r_{pos} : n]$
end if
Carry out the mutation operation on P_1, P_2 in Algorithm 3.
return: P_1 and P_2;

3) Genetic operations: In our paper, we use two genetic operators namely crossover and mutation operations to generate new offspring from the parent individuals. The main process of a single-point crossover algorithm is described in Algorithm 2. First, two-parent individuals G_1 and G_2 are randomly selected from the population of the t-generation. This algorithm begins by generating a random value $r_c \in [0, 1]$. If the r_c is smaller than the predefined crossover probability p_c, the algorithm proceeds with the crossover. Specifically, a random crossover position r_{pos} is selected. Then, the offspring P_1 and P_2, are obtained by swapping the segments between the parents G_1 and G_2 at the r_{pos} single point.

Algorithm 3 Mutation

Input: P: Individual, p_m: Mutation probability, LO_{set}: Local Operators, GO_{set}: Global Operators;
Output: P: Mutated individual;

r_m = Generate a random number between 0 and 1;
if $r_m < p_m$ **then**
  num_pos = Randomly choose from {1, 2, 3}
  positions = Randomly pick *num_pos* positions from P;
  for each position i in *positions* **do**
    if $P[i]$ is a LO operator **then**
      $P[i]$ = Randomly select from LO_{set}
    else
      $P[i]$ = Randomly select from GO_{set}
    end if
  end for
else
  Keep the original individual P;
end if
return: P;

Following the crossover, the mutation operation (as described in Algorithm 3) is applied to the offspring P, which aims to maintain genetic diversity and prevent getting stuck in sub-optimal regions of the solution landscape. The process involves altering one or more genes of a selected individual to explore new regions of the solution space. Concretely, we randomly select 1, 2, or 3 positions within the network configuration. Based on the selected positions, we apply different operators depending on whether the position belongs to the local operator (LO) or the global operator (GO).

IV. Experiments

A. Experimental Settings

1) Datasets: To evaluate the proposed ENAS-TSF, we conducted experiments on three well-established benchmarks: Electricity, Exchange-rate, and ETTh2 following [5], [8], [9]. The detail of experiment datasets are described as follows: 1) Elecitricy dataset inlcudes the electricity consumption of 321 clients with each column corresponding to one client. 2) Exchange-rate shows the current exchange of 8 countries. 3) ETTh2 dataset records hourly from two electricity transformers at two stations.

2) Implementation details: Our method was trained using the L2 loss and optimized with the ADAM optimizer, initialized with a learning rate of 10^{-3}. The batch size was set to 32. The parameters for the Genetic Algorithm were configured as follows: the number of generations is set to

TABLE I: Multivariate forecasting results with different prediction lengths $H \in \{96, 192, 336, 720\}$ and fixed lookback window length $L = 96$. **Red** values indicate the best results, while underlined values indicate the second-best results. Rank 1^{st} refers to the number of times the method performed best.

Model →		**ENAS-TSF (ours)**		**TimesNet** [6]		**FEDformer** [7]		**Crossformer** [8]		**Autoformer** [5]		**Informer** [9]	
Dataset ↓		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Electricity	96	**0.144**	**0.242**	0.169	0.273	0.195	0.309	0.754	0.717	0.204	0.318	0.747	0.652
	192	**0.168**	**0.268**	0.186	0.288	0.202	0.315	0.764	0.721	0.225	0.333	0.877	0.730
	336	**0.186**	**0.286**	0.202	0.302	0.230	0.343	0.777	0.726	0.245	0.349	0.906	0.759
	720	**0.209**	**0.306**	0.232	0.325	0.264	0.367	0.804	0.737	0.298	0.385	0.944	0.793
	Avg	**0.177**	**0.276**	0.197	0.297	0.223	0.334	0.775	0.725	0.243	0.346	0.868	0.733
Exchange-rate	96	**0.088**	**0.207**	0.112	0.240	0.159	0.288	0.274	0.380	0.162	0.292	0.959	0.792
	192	**0.192**	**0.313**	0.218	0.336	0.282	0.386	0.512	0.535	0.336	0.418	1.066	0.835
	336	**0.348**	**0.427**	0.380	0.451	0.435	0.483	0.987	0.769	0.520	0.533	1.613	1.012
	720	**0.913**	**0.720**	0.949	0.741	1.176	0.834	1.534	1.001	1.202	0.849	2.639	1.334
	Avg	**0.385**	**0.417**	0.414	0.442	0.513	0.498	0.827	0.672	0.555	0.523	1.569	0.993
ETTh2	96	**0.304**	**0.350**	0.325	0.368	0.356	0.398	0.859	0.672	0.378	0.412	3.340	1.464
	192	0.432	0.428	**0.418**	**0.421**	0.438	0.444	1.450	0.899	0.460	0.459	5.481	1.959
	336	**0.448**	0.453	0.455	**0.451**	0.466	0.474	2.417	1.282	0.476	0.483	5.162	1.939
	720	0.474	**0.465**	0.465	0.470	**0.457**	0.483	3.276	1.525	0.503	0.504	4.041	1.703
	Avg	**0.414**	**0.424**	0.416	0.428	0.429	0.450	2.000	1.095	0.454	0.464	4.506	1.766
Rank	1^{st}	13	13	1	2	1	0	0	0	0	0	0	0

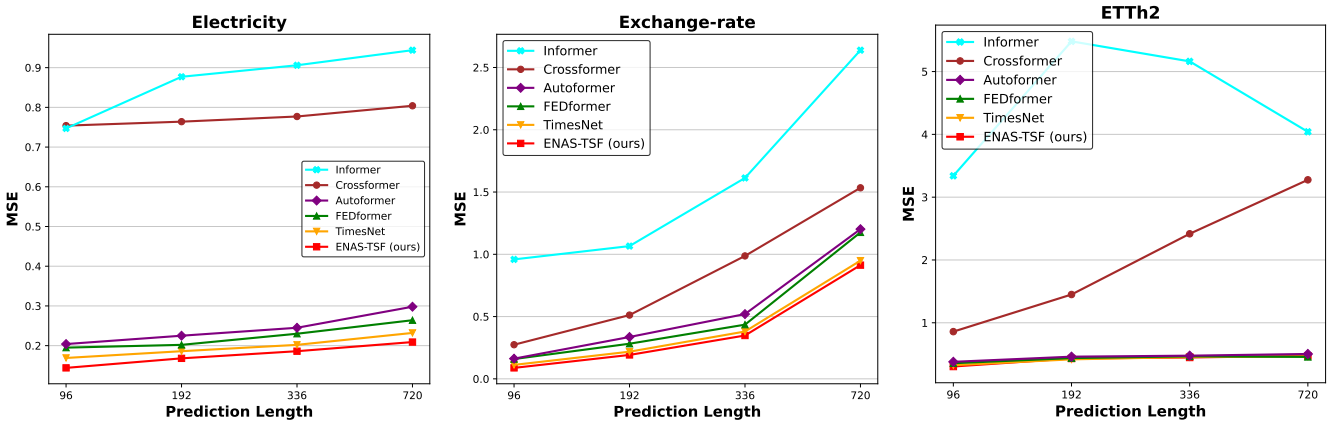


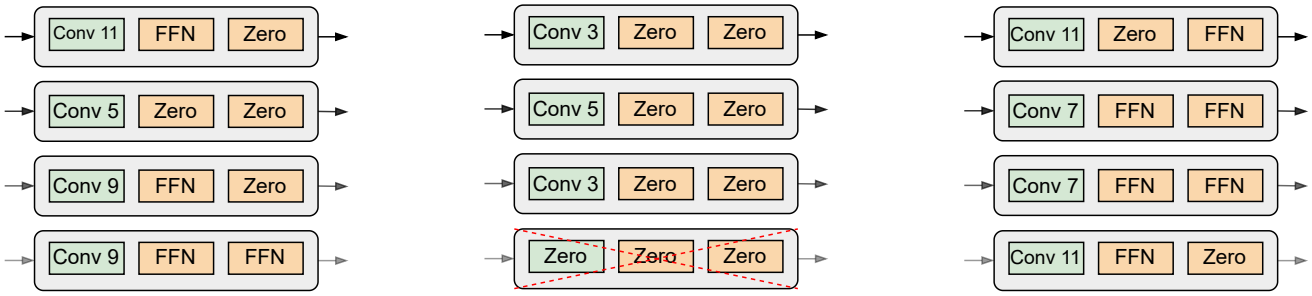
Fig. 2: Influence of prediction length $H \in \{96, 192, 336, 720\}$. ENAS-TSF performs generally better than other models under different prediction lengths.

200, the population size is set to 50, and the probabilities of crossover and mutation are 0.9 and 0.2, respectively. The Genetic Algorithm was early stopped if no improvement in fitness performance was observed within 5 generations.

3) Baseline algorithms: We compared ENAS-TSF with five state-of-the-art multivariate long-term forecasting approaches: Informer [9], Autoformer [5], Crossformer [8], FEDformer [7], and TimesNet [6]. In our experiments, we reproduced the results of these baseline models by replicating the conditions described in their original papers. This involved using the original hyperparameters, runtime environments, and official implementations, including model architectures and code.

B. Results and Discussions

1) Main results: We evaluated the proposed ENAS-TSF framework against state-of-the-art time series forecasting models across three datasets, with prediction horizons $H \in \{96, 192, 336, 720\}$. Comprehensive forecasting results are listed in Table I with the best in **red** and the second-best results marked in underlined. The lower MSE/MAE values indicate more accurate predictions. On the Electricity dataset, ENAS-TSF consistently achieves the lowest MSE and MAE values across all prediction lengths. At $H = 96$, ENAS-TSF obtains an MSE of 0.144 and MAE of 0.242, outperforming TimesNet (MSE: 0.169 and MAE: 0.273) by 14.8% and 11.4%, respectively. Even at the longest horizon ($H = 720$), ENAS-TSF maintains the first rank, with MSE: 0.209 and MAE: 0.306, that is 9.9% (MSE) and 5.8 % (MAE) better than TimesNet. On average, ENAS-TSF is better than the second-best method (TimesNet) by 10.2% for MSE (0.177 and 0.197) and 7.1% for MAE (0.276 and 0.297). These results suggest



(a) Searched on Electricity (b) Searched on Exchange-rate (c) Searched on ETTh2

Fig. 3: The best forecasting architectures found on Electricity, Exchange-rate, and ETTh2 with horizontal forecasting 96. We simplify the connections and different input/output blocks for better observation.

that ENAS-TSF is particularly well-suited to the electricity consumption dataset.

The Exchange-rate dataset exhibits similar trends in which ENAS-TSF outperforms the benchmark algorithms on all forecasting horizons. At $H = 96$, ENAS-TSF dramatically outperforms the second-best method (TimesNet) MSE by 21.4% (0.088 vs. 0.112) and MAE by 13.8% (0.207 and 0.240). At $H = 720$, ENAS-TSF is better than TimesNet by 3.8% for MSE (0.913 vs. 0.949) and by 2.8% for MAE (0.720 vs. 0.741). With the longest horizon $H = 720$, Informer exhibits the poorest performance, with an MSE of 2.639, highlighting its limitations in handling long-horizon forecasting tasks compared to ENAS-TSF. The average MSE (0.385) and MAE (0.417) of ENAS-TSF are 7.0% and 5.7% lower than TimesNet's averages (0.414 and 0.442). While ENAS-TSF secures the best average performance (MSE: 0.414 and MAE: 0.424) on the ETTh2 dataset, minor trade-offs happen at specific horizons. At $H = 192$, TimesNet slightly outperforms our proposed method by 3.2% in MSE (0.418 vs. 0.432) and 1.6% (0.428 vs. 0.421). Meanwhile, at the longest horizon, ENAS-TSF achieves the lowest MAE (0.465 vs. TimesNet's 0.470) despite a slight MSE trade-off (0.474 vs. 0.465).

As demonstrated in Table I, ENAS-TSF yields superior performance compared to various state-of-the-art forecasting methods, securing 13 first-place rankings in both MSE and MAE across all datasets. In contrast, TimesNet ranks first on 1 case for MSE and 2 cases for MAE, while FEDformer obtains 1 first-place result in MSE and none in MAE. The results reveal that Crossformer, Autoformer, and Informer fail to achieve any top-ranked outcomes. As a result, it is concluded that ENAS-TSF model possesses robust performance and broad applicability in multivariate time series forecasting tasks.

2) Effects of different prediction lengths: Fig. 2 illustrates the performance of ENAS-TSF compared to baseline models (Informer, Crossformer, Autoformer, FEDformer, TimesNet) across different prediction lengths $H \in \{96, 192, 336, 720\}$. The figure reveals a clear trend as H increases, the MSE rises across all methods, reflecting the error propagation and growing uncertainty in long-horizon forecasting. ENAS-TSF consistently outperforms baselines, maintaining statistically narrower error margins even at longest prediction $H = 720$. This robustness from its neural architecture search (NAS)-optimized design, which dynamically balances shorter and longer horizon forecasting.

C. Search Result Analysis

1) Architecture Visualization: The Fig. 3 shows the best network architectures discovered by using ENAS-TSF on the Electricity, Exchange-rate, and ETTh2 datasets under a prediction length $H = 96$. We observe that two best-found architectures on Electricity (Fig. 3a) and ETTh2 (Fig. 3c) have highly similar designs, which prefer large-kernel convolutions to capture long-term temporal dependencies. For the global operators, these architectures prefer Feed-Forward Network (FFN) modules and "Zero" blocks. In contrast, the best architecture for Exchange-rate (Fig. 3b) employs small-kernel convolutions. Whenever all operators in a branch are assigned Zero blocks, that branch is effectively pruned, thereby simplifying the overall network. This outcome exemplifies ENAS-TSF's ability to automatically discover dataset-specific architectures by selectively activating or deactivating model components.

2) Model efficiency: The ENAS-TSF requires a longer overall training period than the benchmark algorithms. For example, on the ECL dataset, with a look-back window of $L = 96$ and a forecast horizon of $H = 720$, ENAS-TSF takes about 5 days and 5 hours to find the best solution. This is because the architecture search phase explores and evaluates numerous candidate networks. Meanwhile, purely handcrafted models spend fewer total hours (for example TimesNet with 6.2 hours of training) to reach its final configuration. However, if we examine training speed per iteration, ENAS-TSF operates efficiently and also maintains a relatively small model size. As illustrated in Fig. 4, each forecasting method is visualized according to its model size (circle area), training speed (horizontal axis), and MSE (vertical axis). The figure reveals that ENAS-TSF achieves a lower MSE and training speed than many handcrafted competitors. To the best of our knowledge, this is the first research of NAS to

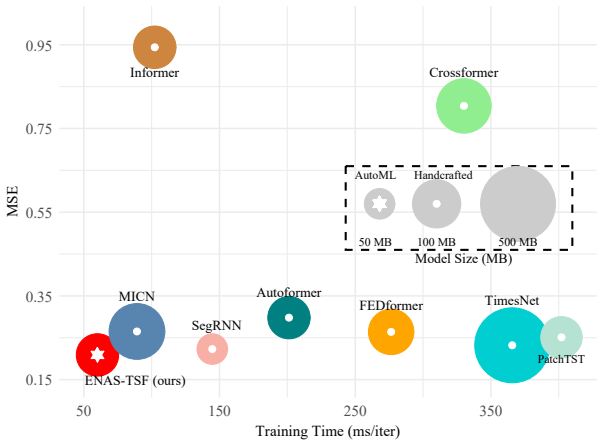


Fig. 4: Model efficiency comparison under input-96-predict-720 of ENAS-TSF and state-of-the-art forecasting models on Electricity dataset.

long-term forecasting tasks. By exploring numerous candidate models and optimizing architectures automatically, ENAS-TSF demonstrates that NAS-based approaches can reduce per-iteration training time and model size compared to handcrafted solutions.

V. Conclusion

In this paper, we introduced the ENAS-TSF framework, a novel approach designed to automate the discovery of optimal deep learning architectures for long-term time series forecasting. By leveraging a multi-branch search space, ENAS-TSF effectively captures both local and global temporal dependencies. Local context modelling can achieve through convolutional operations with different kernel sizes, while global context modelling is offered by attention mechanisms and feed-forward layers. In addition to designing an efficient search space, we employed Genetic Algorithm to efficiently explore and identify the best performing architecture. To enhance computational efficiency, we also proposed an early stopping generation strategy to accelerate the convergence of the search process. Experiments on three multivariate time series forecasting show the effectiveness and robustness of the proposed approach, highlighting its potential for time series forecasting.

Acknowledgment

The authors would like to thank Innovate UK and ADC Energy Ltd for their financial support through the Knowledge Transfer Partnership (KTP) project (No. 13518).

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[2] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: John Wiley & Sons, 2015.

[3] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Melbourne, Australia: OTexts, 2018. [Online]. Available: https://otexts.com/fpp2/

[4] A. Vaswani, *et al.*, "Attention is all you need," in *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.

[5] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 22419–22430.

[6] H. Wu, J. Xu, J. Wang, and M. Long, "TimesNet: Temporal 2D-variation modeling for general time series analysis," 2022, arXiv:2210.02186. [Online]. Available: https://arxiv.org/abs/2210.02186

[7] T. Zhou, *et al.*, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 27268–27286.

[8] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *Proc. 11th Int. Conf. Learn. Represent. (ICLR)*, 2023.

[9] H. Zhou, *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 11106–11115.

[10] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," 2019, arXiv:1905.10437. [Online]. Available: https://arxiv.org/abs/1905.10437

[11] A. Zeng, M. Chen, L. Zhang, Q. Xu, "Are transformers effective for time series forecasting?," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 9, 2023, pp. 11121–11128.

[12] A. Gulati, *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," 2020, arXiv:2005.08100. [Online]. Available: https://arxiv.org/abs/2005.08100

[13] Y. Wang, *et al.*, "Deep time series models: A comprehensive survey and benchmark," *arXiv* 2024, arXiv:2407.13278. [Online]. Available: https://arxiv.org/abs/2407.13278

[14] W. Cao, *et al.*, "Brits: Bidirectional recurrent imputation for time series," in *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[15] G. Lai, W. C. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *Proc. 41st Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, 2018, pp. 95–104.

[16] H. Wang, *et al.*, "Micn: Multi-scale local and global context modeling for long-term series forecasting," in *Proc. 11th Int. Conf. Learn. Represent. (ICLR)*, 2023.

[17] D. Luo and X. Wang, "Moderntcn: A modern pure convolution structure for general time series analysis," in *Proc. 12th Int. Conf. Learn. Represent. (ICLR)*, 2024.

[18] Z. Wu, *et al.*, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2020, pp. 753–763.

[19] T. H. Vu, *et al.*, "FAT: Fusion-attention transformer for remaining useful life prediction," in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, Springer, Cham, 2025, pp. 286–301.

[20] E. Real, *et al.*, "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2902–2911.

[21] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 1379–1388.

[22] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, 2019, pp. 394–407.

[23] Z. Liang and Y. Sun, "Evolutionary neural architecture search for multivariate time series forecasting," in *Proc. Asian Conf. Mach. Learn. (ACML)*, 2024, pp. 771–786.

[24] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 35, 2022, pp. 9881–9893.

[25] O. Press, N. A. Smith, and O. Levy, "Improving transformer models by reordering their sublayers," *arXiv* 2019, arXiv:1911.03864. [Online]. Available: https://arxiv.org/abs/1911.03864

[26] J. L. Ba, J R. Kiros, G. E. Hinton, "Layer normalization," *arXiv* 2016, arXiv:1607.06450. [Online]. Available: https://arxiv.org/abs/1607.06450