AutoMixer: Automated Multi-Scale Mixing for Time Series Forecasting Based on Neural Architecture Search

Jiangrong Yang†
Beijing Normal University
Beijing, China
jiangrongyang@mail.bnu.edu.cn

Haodi Wang†
City University of Hong Kong
Hong Kong SAR, China
haodi.wang@cityu.edu.hk

Tangyu Jiang
Hong Kong Baptist University
Hong Kong SAR, China
tyjiang@hkbu.edu.hk

Maiyun Zhang
Beijing Normal University
Beijing, China
maiyunzhang@mail.bnu.edu.cn

Rongfang Bie*
Beijing Normal University
Beijing, China
rfbie@bnu.edu.cn

***Abstract*—Time series forecasting has become an emerging paradigm for predicting future temporal variations. Real-world time series data exhibits heterogeneous fluctuations at different scales, thus demonstrating multi-scale characteristics. Previous works either rely heavily on the manual design of the feature-mixing architecture or suffer from the high computational cost during the search procedure. Moreover, the performance of the existing methods can be further enhanced. To address the above issues, in this paper, we propose an automated multi-scale mixing approach for time series forecasting based on the Neural Architecture Search (NAS) called AutoMixer. Unlike the previous works, AutoMixer can automatically generate the adaptive fusion architecture for both multivariate and single-variate datasets. By proposing a customized search space and concrete training algorithm, AutoMixer realizes an efficient multi-scale feature-mixing without invoking extra expert knowledge. The experimental results show that AutoMixer outperforms the state-of-the-art methods with 6 to 70 times faster search time compared to other NAS-based schemes with state-of-the-art accuracy. For instance, our method achieves 11.98 of MAPE on PEMS04, which is 4.3% better than the best baseline.**

***Index Terms*—Time Series, Time Series Forecast, Neural Architecture Search**

I. Introduction

Time series forecasting has become a promising paradigm due to its ability to predict future temporal variations given past time observations. The effectiveness makes it essential for various applications including traffic planning, weather prediction or electricity forecasting [1].

The existing methods are established by crafting the structures of deep learning models. The most fundamental methods rely on CNNs to extract the sub-series features [2], RNNs to model the temporal dependency [3], and GNNs for correlated time series [4]. Another line of work is constructed based on the Transformer models, which have been extensively adopted in time series forecasting recently [5]–[9].

Among all the previous methods, the time series data is proven to have the characteristic of multi-scale. More concretely, time series in the real world usually exhibit heterogeneous fluctuations at various scales. For instance, household electricity usage exhibits scale-dependent patterns: minute-level spikes from appliances, daily evening peaks, and monthly seasonal fluctuations driven by heating or cooling.

To capture and utilize this trait, numerous multi-scale modeling approaches are proposed for the time series forecasting tasks [2], [7], [10]–[12]. In 2024, [13] proposed TimeMixer, in which the authors designed an MLP-based model with feature decomposition and mixing blocks. TimeMixer achieves state-of-the-art performance in both long-term and short-term tasks on multiple datasets.

Though effective, the majority of the existing works rely highly on the manual design of the network architectures. For instance, TimeMixer proposes a multi-scale fusion structure as its core component which realizes efficient time series forecasting with high accuracy. However, its fusion methods are designed manually (i.e., bottom-up or top-down), which requires extra expert knowledge and may not be the optimal architecture. In fact, manually designing the fusion model suffers from time-consuming trial-and-error and might introduce bias into the method. The rigid architectures are also prohibitive in adapting to various characteristics of different datasets. Although there are a few works proposed as automated correlation time series methods [14]–[18], they cannot be trivially adapted for the multi-scale decomposition and fusion. The complicated search spaces also incur a high time cost. Furthermore, the performance of the existing time series forecasting schemes can be further improved.

Our Objectives. To address the above issues, the main objective of this paper is to shed some light on the automated multi-scale mixing algorithms for time series forecasting. In contrast to the previous work, our proposed method should

†Equal Contribution. *Corresponding Author.

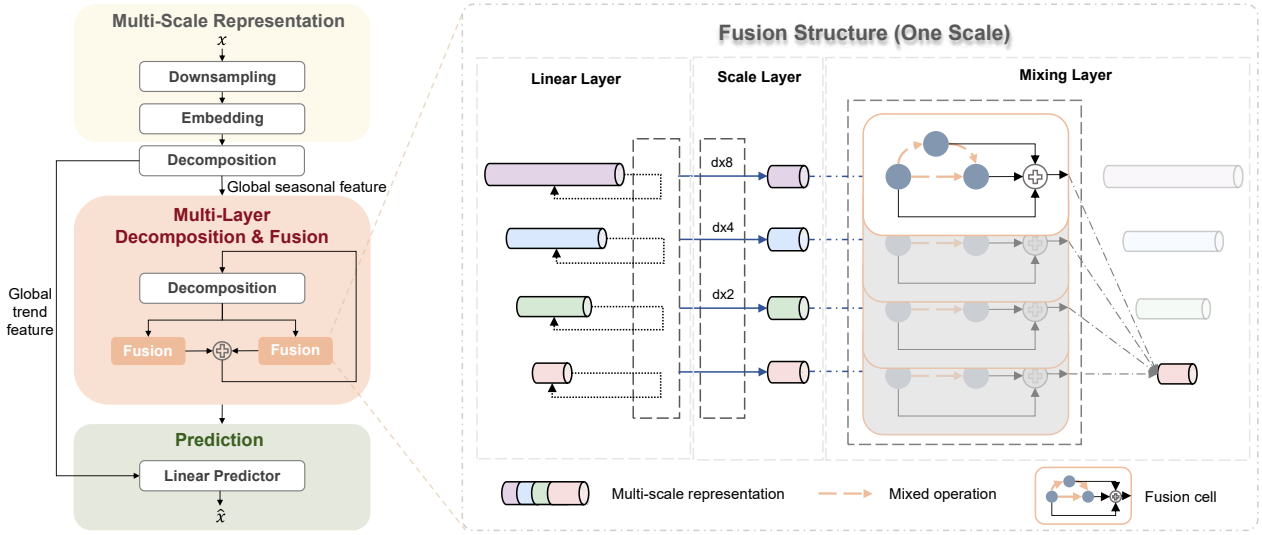

Fig. 1: Framework overview of AutoMixer.

be efficient in automatically designing of the fusion structure, which can generate superior mixing architectures adaptive to various datasets. Also, our approach should achieve better performance than the state-of-the-art methods.

Our Contributions. To this end, in this paper, we propose a novel and automated multi-scale mixing method for time series forecasting based on the Neural Architecture Search (NAS). The proposed scheme, called AutoMixer, leverages a one-shot NAS framework to achieve adaptive multi-scale feature decomposition and fusion. At its core, the proposed decomposition-fusion module automatically generates optimal mixing architectures through a rigorously designed search space of customized operations. Compared to handcrafted and automated baselines, AutoMixer demonstrates superior performance with significantly reduced execution time. Specifically, on the PEMS04 dataset, AutoMixer improves prediction accuracy by over 2% across all metrics. Furthermore, it completes the search in 0.67 GPU hours—over 10× faster than the state-of-the-art NAS model AutoCTS+ [17].

In all, the concrete contributions are summarized as follows:

- We propose AutoMixer, a multi-scale mixing framework for time series forecasting that uses NAS to eliminate the manual design of the architecture. Compared to the handcrafted methods, AutoMixer avoids the time-consuming trial-and-error procedure and realizes adaptive fusion for various datasets.
- We design a customized and effective search space for the multi-scale fusion module, which achieves a speedup of 6 to 70 times over previous NAS methods for time series prediction.
- We rigorously implement our AutoMixer and extensively design the experiments to evaluate its performance on several widely used real-world time series datasets. The experimental results demonstrate that our approach can discover superior fusion architectures and achieve state-of-the-art performance in prediction tasks. Our code will be open-sourced and available.

II. Methodologies

A. Framework Overview

The framework of AutoMixer is illustrated in Figure 1, consisting of three components: the feature representation module, the multi-layer decomposition and fusion module, and the prediction module.

Multi-Scale Feature Representation. To capture both local patterns and coarse variations [19], the input $\mathbf{x} \in \mathbb{R}^{N \times T}$ is represented across K scales via downsampling. Let $\mathbf{x}_0 = \mathbf{x}$ and $\mathbf{x}_k = \mathsf{AvgPool}(\mathbf{x}_{k-1})$ for $k \in \{1, \ldots, K\}$, yielding $\mathbf{x}_k \in \mathbb{R}^{N \times \lceil T/2^k \rceil}$. The resulting set is $\mathcal{X} = \{\mathbf{x}_0, \ldots, \mathbf{x}_K\}$. Each scale is then normalized and embedded:

$$\mathbf{x}'_k = \mathsf{Embed}(\mathsf{Norm}(\mathbf{x}_k)), \quad k \in 0, \ldots, K \tag{1}$$

where $\mathsf{Norm}(\cdot)$ denotes z-score normalization and $\mathsf{Embed}(\cdot)$ is a 1D convolution mapping the N channels to a d_{model}-dimensional space, producing the representations $\mathcal{X}' = \{\mathbf{x}'_0, \ldots, \mathbf{x}'_K\}$.

Multi-Layer Decomposition and Fusion. This module performs cross-scale feature mixing after decomposition and serves as the core of AutoMixer. Neural Architecture Search (NAS) is employed to automatically determine the multi-scale fusion structure. The output is the multi-scale representation $\mathcal{X}^M = \{\mathbf{x}_0^M, \ldots, \mathbf{x}_K^M\}$.

Prediction. The final prediction $\hat{\mathbf{x}} \in \mathbb{R}^{N \times T_{out}}$ is obtained by aggregating the seasonal representations $\mathcal{X}^M$ with the global trend components $\mathcal{T}$. For each scale k, linear predictors map features to the output length:

$$\hat{\mathbf{x}} = \mathsf{Norm}\left(\sum_{k=0}^{K} \mathsf{Rec}\left(\mathsf{Pred}(\mathbf{x}_k^M) + \mathsf{Pred}(\mathsf{FC}(\mathbf{t}_k))\right)\right) \tag{2}$$

where $\mathsf{FC}(\cdot)$ aligns feature dimensions, $\mathsf{Pred}(\cdot)$ projects the temporal dimension to T_{out}, and $\mathsf{Rec}(\cdot)$ maps the latent dimension d_{model} back to N channels.

B. Multi-Layer Decomposition and Fusion

1) Overview: As shown in Figure 1, the multi-scale representation $\mathcal{X}'$ is first decomposed into global seasonal and trend components:

$$\mathcal{S}, \mathcal{T} = \mathsf{Decompose}(\mathcal{X}') \tag{3}$$

The trend features $\mathcal{T}$ are fed into the prediction module, while the seasonal features $\mathcal{S}$ are used as the input of the first DF layer. In each layer, the input is decomposed into seasonal and trend components and fused separately across scales. Their outputs are summed to form the next-layer representation. Neural Architecture Search (NAS) is applied to automatically determine the fusion structure.

Formally, the m-th layer is defined as:

$$\begin{aligned} \mathcal{X}_s^{m-1}, \mathcal{X}_t^{m-1} &= \mathsf{Decompose}(\mathcal{X}^{m-1}) \\ \mathcal{X}^m &= \mathsf{Fusion}(\mathcal{X}_s^{m-1}) + \mathsf{Fusion}(\mathcal{X}_t^{m-1}) \end{aligned} \tag{4}$$

where $\mathcal{X}^m = \{\mathbf{x}_0^m, \ldots, \mathbf{x}_K^m\}$ and M denotes the number of DF layers.

2) Decomposition: Time series usually contain seasonal fluctuations and long-term trends. Following previous works [13], [20], [21], we decompose each scale representation using a moving-average operation:

$$\begin{aligned} \mathbf{x}_{k,t}^m &= \mathsf{AvgPool}(\mathsf{Padding}(\mathbf{x}_k^m)) \\ \mathbf{x}_{k,s}^m &= \mathbf{x}_k^m - \mathbf{x}_{k,t}^m \end{aligned} \tag{5}$$

where $\mathbf{x}_{k,s}^m$ and $\mathbf{x}_{k,t}^m$ denote the seasonal and trend terms. We denote the collections of these components as $\mathcal{X}_t^m = \{\mathbf{x}_{0,t}^m, \ldots, \mathbf{x}_{K,t}^m\}$ and $\mathcal{X}_s^m = \{\mathbf{x}_{0,s}^m, \ldots, \mathbf{x}_{K,s}^m\}$, which are sent to the fusion block.

3) Fusion Structure: The fusion block contains three stages: a linear layer, a scale alignment layer, and a NAS-based mixing layer. Since the structures for trend and seasonal features are identical, we denote $\mathcal{X}_D^m = \{\mathbf{x}_{0,D}^m, \ldots, \mathbf{x}_{K,D}^m\}, D \in \{s, t\}$.

Linear Layer. To model inter-variable dependencies before fusion, each scale is transformed by two linear layers with GELU activation:

$$\mathbf{u}_{k,D}^m = \mathsf{Linear}_2(\mathsf{GELU}(\mathsf{Linear}_1(\mathbf{x}_{k,D}^m))) \tag{6}$$

Scale Layer. Since the representation lengths differ across scales, we align them to the k-th scale before mixing:

$$\mathbf{v}_{k',D}^m = \begin{cases} \mathsf{Up}_{\mathsf{ur}}(\mathbf{u}_{k',D}^m) & k' > k \\ \mathbf{u}_{k',D}^m & k' = k \\ \mathsf{Down}_{\mathsf{dr}}(\mathbf{u}_{k',D}^m) & k' < k \end{cases} \tag{7}$$

This operation resamples features so that all inputs share the same temporal dimension.

Mixing Layer. The aligned representations are fused across scales using a NAS-based supernet:

$$\mathbf{x}_{k,D}^m = \sum_{k'=0}^{K} \mathsf{Cell}_{k'}(\mathbf{v}_{k',D}^m) \tag{8}$$

Each cell is a DAG with nodes $\mathbf{z}_{k'}$ and candidate operations $\mathcal{O} = \{\textit{zero}, \textit{identity}, \mathsf{Linear}\}$. Edges are weighted by architecture parameters α and relaxed with softmax during search:

$$\mathbf{z}_{k'}^j = \sum_{i<j} \sum_{Op \in \mathcal{O}} \frac{\exp(\alpha_{Op}^{(i,j)})}{\sum_{Op'} \exp(\alpha_{Op'}^{(i,j)})} \mathsf{Op}(\mathbf{z}_{k'}^i) \tag{9}$$

After search, the strongest operation on each edge is selected:

$$\mathbf{z}_{k'}^{h_{k'}} = \mathbf{z}_{k'}^0 + \sum_{i<j} \mathsf{argmax}_{Op \in \mathcal{O}} \alpha_{Op}^{(i,j)}(\mathbf{z}_{k'}^i) \tag{10}$$

This NAS design enables automatic multi-scale feature fusion with lower search cost than previous NAS-based forecasting models.

C. Training Algorithm

During the training of AutoMixer, we adopt the stochastic gradient descent algorithm to update the weight parameters of the training set and the architecture parameters of the validation set. The goal of the training is to minimize the loss on the validation set when fixing the parameters of the training data. We alternatively update the architecture and model weight parameters to avoid mutual impact between these two parts. To facilitate the process, we adopt the settings in [30] and use the first-order approximation, such that

$$\begin{aligned} &\min_{\Lambda} \mathcal{L}_{valid}(\Theta^*(\Lambda), \Lambda), \\ &\text{s.t. } \Theta^*(\Lambda) = \arg\min_{\Theta} \mathcal{L}_{train}(\Theta, \Lambda), \end{aligned} \tag{11}$$

where Θ denotes the weight parameters of the model and Λ denotes the architecture parameters, and Θ^* denotes the optimal weight parameters. $\mathcal{L}_{valid}$ and $\mathcal{L}_{train}$ refer to the loss at the validation set and training set, respectively.

III. Experiments and Evaluations

In this section, we aim to answer the following key questions:

- What is the performance gain of AutoMixer on various datasets and tasks compared to baselines? (Section III-A)
- How fast is AutoMixer compared to the NAS-based methods? (Section III-B)
- What is the impact of each module in AutoMixer ? (Section III-C)

To this end, we conduct comprehensive experiments on five commonly used benchmark datasets for time series forecasting, including two short-term forecasting datasets: PEMS datasets and M4 dataset (with six subsets), and three long-term forecasting datasets: ETT datasets (with four subsets, i.e., ETTh1, ETTh2, ETTm1, and ETTm2), Weather datasets,

TABLE I: Performance metrics for different models on PEMS datasets.

Model \ Data	PEMS03			PEMS04			PEMS07			PEMS08		
	MAE	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**	**RMSE**
Informer [5]	19.19	19.58	32.70	22.05	14.88	36.20	27.26	11.63	45.81	20.96	13.20	30.61
Autoformer [20]	18.08	18.75	27.82	25.00	16.70	38.02	26.92	11.83	40.60	20.47	12.27	31.52
Stationary [22]	17.64	17.56	28.37	22.34	14.85	35.47	26.02	11.75	42.34	19.29	12.21	38.62
FEDformer [6]	19.00	18.57	30.05	26.51	16.76	41.81	27.92	12.29	42.29	20.56	12.41	32.97
DLinear [21]	19.70	18.35	32.35	24.62	16.12	39.51	28.65	12.15	45.02	20.26	12.09	32.38
FiLM [23]	21.36	18.35	35.07	26.74	16.46	42.86	28.76	11.21	45.85	22.11	12.81	35.13
MICN [2]	15.71	15.67	24.55	21.62	13.53	34.39	22.28	9.57	35.40	17.76	10.76	27.26
TimesNet [24]	16.41	15.17	26.72	21.63	13.15	34.90	25.12	10.60	40.71	19.01	11.83	30.65
PatchTST [8]	18.95	17.29	30.15	24.86	16.65	40.46	27.87	12.69	42.56	20.35	13.15	31.04
Crossformer [7]	15.64	15.74	25.56	20.38	12.84	32.41	22.54	9.38	35.49	17.56	10.92	27.21
SCINet [12]	15.97	15.89	25.20	20.35	12.84	32.31	22.79	9.41	35.61	17.38	10.80	27.34
TiDE [25]	18.39	17.39	30.59	23.99	14.76	37.30	25.31	11.02	39.12	19.73	12.02	30.67
iTransformer [26]	16.72	15.81	27.81	21.81	13.42	33.91	23.01	10.02	35.56	17.94	10.93	27.88
TimeMixer [13]	<u>14.63</u>	**14.54**	<u>23.28</u>	<u>19.21</u>	<u>12.53</u>	<u>30.92</u>	<u>20.57</u>	<u>8.62</u>	<u>33.59</u>	15.22	<u>9.67</u>	24.26
AutoSTG [15]	16.49	15.14	26.43	21.72	13.99	34.83	23.45	9.81	37.77	17.37	10.83	27.87
AutoCTS [16]	16.36	15.53	23.96	21.60	15.82	32.36	23.97	10.67	36.64	17.20	12.46	26.41
AutoCTS+ [17]	15.22	14.71	24.68	19.40	13.99	30.80	22.92	9.79	36.14	**15.12**	9.68	**24.00**
AutoMixer **(Ours)**	**14.41**	**14.60**	**23.09**	**18.56**	**11.98**	**30.30**	**20.50**	**8.49**	**33.35**	**15.14**	**9.65**	**24.23**

and Solar-Energy datasets. We compare the performance of AutoMixer with over 19 SOTA baselines in terms of MAE, MAPE, and RMSE.

Our code is open-sourced and available at

https://github.com/delight16/AutoMixer

A. *Performance of* AutoMixer

The results for both short-term and long-term forecasting are reported in Tables I, II, and III, where the best results are in bold and the second-best are underlined.

Short-Term Forecasting. Table I shows the performance on PEMS datasets with input length 96 and output length 12. Overall, AutoMixer achieves the best results on most metrics compared with handcrafted and NAS-based baselines. For example, on PEMS03 the MAE decreases from 19.19 (Informer) to 14.41, improving by 4.78. Compared with the multi-scale method TimeMixer, AutoMixer improves MAE by about 3%.

The results on the M4 dataset are presented in Table II. AutoMixer achieves competitive or better performance than existing methods. Meanwhile, due to the simplified search space for multi-scale features, the search cost of AutoMixer is significantly lower than previous NAS-based approaches.

Long-Term Forecasting. Table III reports results using input length 336 and prediction horizons {96, 192, 336, 720}. Our method achieves the best performance on most tasks. Compared with automatic methods (AutoSTG and AutoCTS+), AutoMixer shows improvements of up to 2.7 and better stability across datasets. It is also competitive with the SOTA handcrafted model TimeMixer, which achieves only slightly lower MAE (0.3%) and MSE (0.6%).

B. *Efficiency Analysis*

We compare the search time against the NAS-based models in Table IV, with all experiments conducted on an NVIDIA GeForce RTX 3090 GPU (24GB RAM) for a fair comparison.

AutoMixer achieves a $6\times$ to $70\times$ speedup in search time on all datasets of PEMS than the baselines. For instance, AutoMixer only costs 2.05, 0.67, 2.98, and 1.36 hours on PEMS03, PEMS04, PEMS07, and PEMS08, respectively, while AutoSTG requires 57.17, 20.32, 129.76, and 21.93 hours, which is approximately 20 times slower than ours. Note that the training of the comparator in AutoCTS+ is non-trivial due to the noisy level of the noise dataset and the high training complexity of the clean set. Thus, AutoCTS+ requires considerable upfront time to train the comparator, which is not considered as the search time. In all, the experimental results demonstrate that AutoMixer incurs significantly less search time in comparison with the previous NAS-based methods. These results indicate that our customized search space can effectively reduce the search time while maintaining the optimal prediction performance.

TABLE II: Performance metrics for different models on M4 datasets.

Data / Model	M4-Yearly			M4-Quarterly			M4-Monthly			M4-other			M4-Average		
	SMAPE	**MASE**	**OWA**	**SMAPE**	**MASE**	**OWA**	**SMAPE**	**MASE**	**OWA**	**SMAPE**	**MASE**	**OWA**	**SMAPE**	**MASE**	**OWA**
Informer [5]	14.727	3.418	0.881	11.360	1.401	1.027	14.062	1.141	1.024	24.460	20.960	5.879	14.086	2.718	1.230
Pyraformer [10]	15.530	3.711	0.942	15.449	2.350	1.558	17.642	1.913	1.511	24.786	18.581	5.538	16.987	3.265	1.480
Autoformer [20]	13.974	3.134	0.822	11.338	1.365	1.012	13.958	1.103	1.002	5.485	3.865	1.187	12.909	1.771	0.939
Stationary [22]	13.717	3.078	0.807	10.958	1.325	0.981	13.917	1.097	0.998	6.302	4.064	1.304	12.780	1.756	0.930
FEDformer [6]	13.728	3.048	0.803	10.792	1.283	0.958	14.260	1.102	1.012	4.954	3.264	1.036	12.840	1.701	0.918
DLinear [21]	16.965	4.283	1.058	12.145	1.520	1.106	13.514	1.037	0.956	6.709	4.953	1.487	13.639	2.095	1.051
LightTS [27]	14.247	3.109	0.827	11.364	1.328	1.000	14.014	1.053	0.981	15.880	11.434	3.474	13.525	2.111	1.051
FiLM [23]	17.431	4.043	1.042	12.925	1.664	1.193	15.407	1.298	1.144	7.134	5.09	1.553	14.863	2.207	1.125
MICN [2]	25.022	7.162	1.667	15.214	1.963	1.407	16.943	1.442	1.265	41.985	62.734	14.313	19.638	5.947	2.279
PatchTST [8]	16.463	3.967	1.003	10.644	1.278	0.949	13.399	1.031	0.949	6.558	4.511	1.401	13.152	1.945	0.998
SCINet [12]	18.605	4.471	1.132	14.871	2.054	1.424	14.925	1.131	1.027	16.655	15.034	4.123	15.542	2.816	1.309
N-BEATS [28]	13.436	3.043	0.794	10.124	1.169	0.886	12.677	0.937	0.880	4.925	3.391	1.053	11.851	1.559	0.855
N-HiTS [29]	13.418	3.045	0.793	10.202	1.194	0.899	12.791	0.969	0.899	5.061	3.216	1.040	11.927	1.613	0.861
TiDE [25]	15.32	3.54	0.91	11.83	1.41	1.05	15.18	1.19	1.09	6.12	4.33	1.33	13.95	1.94	1.02
iTransformer [26]	13.92	3.214	0.83	10.76	1.283	0.956	13.80	1.083	0.897	5.57	3.94	1.207	12.68	1.76	0.929
TimesNet [24]	13.387	2.996	0.786	10.100	1.182	0.890	12.670	0.933	0.878	4.891	3.302	1.035	11.829	1.585	0.851
TimeMixer [13]	13.206	**2.916**	0.776	9.996	1.166	**0.825**	12.605	0.919	0.869	4.564	**3.115**	0.982	11.723	1.559	0.840
AutoSTG [15]	13.350	3.019	0.788	11.098	1.453	1.034	13.864	1.112	1.003	10.858	12.561	3.124	12.932	2.205	1.053
AutoCTS [16]	20.940	4.899	1.257	18.535	2.504	1.755	18.221	1.577	1.373	9.463	7.574	2.190	18.484	2.863	1.430
AutoCTS+ [17]	15.096	3.351	0.883	12.943	1.581	1.164	15.973	1.346	1.186	8.881	7.283	2.083	14.689	2.160	1.106
AutoMixer **(Ours)**	**13.181**	**2.968**	**0.775**	**9.953**	**1.164**	**0.876**	**12.532**	**0.915**	**0.864**	**4.542**	**3.117**	**0.969**	**11.663**	**1.557**	**0.837**

TABLE III: Performance metrics for long-term forecasting.

Data / Model		**AutoMixer (Ours)**		**AutoCTS+** [17]		**AutoSTG** [15]		**TimeMixer** [13]		**TimesNet** [24]	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	**Avg**	**0.227**	**0.265**	0.641	0.611	0.289	0.372	0.229	0.277	0.243	0.286
Solar	**Avg**	**0.210**	**0.277**	0.244	**0.268**	0.229	0.289	0.218	0.277	0.243	0.291
ETTh1	**Avg**	**0.445**	**0.447**	1.093	0.810	0.794	0.644	0.476	0.465	0.483	0.478
ETTh2	**Avg**	**0.370**	**0.407**	3.061	1.340	0.733	0.551	**0.370**	0.410	0.415	0.444
ETTm1	**Avg**	**0.359**	**0.386**	0.741	0.639	0.718	0.581	0.373	0.405	0.423	0.432
ETTm2	**Avg**	**0.265**	**0.323**	3.454	1.446	0.386	0.431	0.267	**0.323**	0.283	0.332

TABLE IV: Search time of NAS-based methods in GPU hours.

Model	PEMS03	PEMS04	PEMS07	PEMS08
AutoSTG	57.17	20.32	129.76	21.93
AutoCTS	139.6	56.17	245.62	59.66
AutoCTS+	12.65	7.57	20.72	8.14
AutoMixer **(Ours)**	**2.05**	**0.67**	**2.98**	**1.36**

C. Ablation Study

We use PEMS to analyze the contributions of each component in AutoMixer to the prediction results. Table V presents the ablation results evaluating the contribution of each component in AutoMixer. The full model consistently achieves the best performance across all metrics on four datasets, confirming the effectiveness of all modules.

Removing the multi-layer DF module causes the largest performance drop, highlighting its importance in capturing multi-scale temporal patterns. Within the DF module, the decomposition and fusion components contribute similarly to the overall performance. Replacing the prediction module also degrades performance, though the impact is less significant than the other ablations.

IV. Conclusion

In this paper, We propose AutoMixer, a NAS-based multi-scale mixing framework that automates feature fusion for time

TABLE V: Results of AutoMixer in the ablation experiments

Model		**Ours**	w/o Decomp.	w/o Fusion	w/o Prediction	w/o DF
PEMS03	MAE	**14.41**	15.53	15.33	14.79	16.09
	MAPE	**14.60**	15.89	15.51	14.91	16.91
	RMSE	**23.09**	24.45	24.16	23.60	25.15
PEMS04	MAE	**18.56**	19.58	19.38	18.65	20.39
	MAPE	**11.98**	12.91	12.65	12.01	13.42
	RMSE	**30.30**	31.31	31.16	30.38	32.46
PEMS07	MAE	**20.50**	21.83	21.43	20.57	23.27
	MAPE	**8.49**	9.16	9.15	8.50	10.05
	RMSE	**33.35**	34.58	34.03	33.47	36.03
PEMS08	MAE	**15.14**	15.91	15.74	15.31	16.82
	MAPE	**9.65**	10.15	9.98	9.78	10.55
	RMSE	**24.23**	24.93	24.77	24.55	26.27

series forecasting, eliminating the need for manual trial-and-error. By introducing a customized search space for the fusion cell, AutoMixerachieves a 6–70× speedup over existing NAS methods. Extensive experiments on benchmark datasets show that AutoMixerconsistently discovers superior architectures and achieves state-of-the-art performance, highlighting the potential of automated NAS in enhancing forecasting accuracy and efficiency.

ACKNOWLEDGMENT

The work was supported by National Key R&D Program of China [Grant No. 2022ZD0115901], in part by the National Natural Science Foundation of China Project under Grants No. 62177007.

REFERENCES

[1] M. Blum and M. Riedmiller, "Electricity demand forecasting using gaussian processes," in *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[2] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "Micn: Multi-scale local and global context modeling for long-term series forecasting," in *The eleventh international conference on learning representations*, 2023.

[3] G. Lai, W.-C. Chang, Y. Yang, and H. 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*, 2018, pp. 95–104.

[4] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 753–763.

[5] H. Zhou, S. Zhang, J. Peng, and S. e. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[6] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *International conference on machine learning*. PMLR, 2022, pp. 27 268–27 286.

[7] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The eleventh international conference on learning representations*, 2023.

[8] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *The Eleventh International Conference on Learning Representations*, 2023.

[9] X. Piao, Z. Chen, T. Murayama, Y. Matsubara, and Y. Sakurai, "Fredformer: Frequency debiased transformer for time series forecasting," in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, 2024, pp. 2400–2410.

[10] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International conference on learning representations*, 2021.

[11] M. A. Shabani, A. H. Abdi, L. Meng, and T. Sylvain, "Scaleformer: Iterative multi-scale refining transformers for time series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[12] M. Liu, A. Zeng, M. Chen, Z. Xu, Q. Lai, L. Ma, and Q. Xu, "Scinet: Time series modeling and forecasting with sample convolution and interaction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5816–5828, 2022.

[13] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU, "Timemixer: Decomposable multiscale mixing for time series forecasting," in *The Twelfth International Conference on Learning Representations*, 2024.

[14] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, and Y. Zheng, "Autost: Efficient neural architecture search for spatio-temporal prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 794–802.

[15] Z. Pan, S. Ke, X. Yang, Y. Liang, Y. Yu, J. Zhang, and Y. Zheng, "Autostg: Neural architecture search for predictions of spatio-temporal graph," in *Proceedings of the Web Conference 2021*, 2021, pp. 1846–1855.

[16] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, "Autocts: Automated correlated time series forecasting," *Proceedings of the VLDB Endowment*, vol. 15, no. 4, pp. 971–983, 2021.

[17] X. Wu, D. Zhang, M. Zhang, C. Guo, B. Yang, and C. S. Jensen, "Autocts+: Joint neural architecture and hyperparameter search for correlated time series forecasting," *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–26, 2023.

[18] X. Wu, X. Wu, B. Yang, L. Zhou, C. Guo, X. Qiu, J. Hu, Z. Sheng, and C. S. Jensen, "Autocts++: zero-shot joint neural architecture and hyperparameter search for correlated time series forecasting," *The VLDB Journal*, vol. 33, no. 5, pp. 1743–1770, 2024.

[19] M. A. Ferreira, D. M. Higdon, H. K. Lee, and M. West, "Multi-scale and hidden resolution time series models," 2006.

[20] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22 419–22 430, 2021.

[21] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 11 121–11 128.

[22] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9881–9893, 2022.

[23] T. Zhou, Z. Ma, Q. Wen, L. Sun, T. Yao, W. Yin, R. Jin *et al.*, "Film: Frequency improved legendre memory model for long-term time series forecasting," *Advances in neural information processing systems*, vol. 35, pp. 12 677–12 690, 2022.

[24] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in *The Eleventh International Conference on Learning Representations*, 2023.

[25] A. Das, W. Kong, A. Leach, S. Mathur, R. Sen, and R. Yu, "Long-term forecasting with tide: Time-series dense encoder," *arXiv preprint arXiv:2304.08424*, 2023.

[26] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," *arXiv preprint arXiv:2310.06625*, 2023.

[27] D. Campos, M. Zhang, B. Yang, T. Kieu, C. Guo, and C. S. Jensen, "Lightts: Lightweight time series classification with adaptive ensemble distillation," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–27, 2023.

[28] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations*, 2020.

[29] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski, "Nhits: Neural hierarchical interpolation for time series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 6, 2023, pp. 6989–6997.

[30] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.