

EMTSF: Extraordinary Mixture of SOTA Models for Time Series Forecasting

Musleh Alharthi^{a,*}, Kaleel Mahmood^b, Sarosh Patel^a and Ausif Mahmood^a

^aComputer Science and Engineering, University of Bridgeport, Bridgeport, CT, USA

^bDepartment of Computer Science and Statistics, University of Rhode Island, Kingston, RI, USA

Abstract. The immense success of the Transformer architecture in Natural Language Processing has led to its adoption in Time Series Forecasting (TSF), where superior performance has been shown. However, a recent important paper questioned their effectiveness by demonstrating that a simple single layer linear model outperforms Transformer-based models. This was soon shown to be not as valid, by a better transformer-based model termed PatchTST. More recently, TimeLLM demonstrated even better results by repurposing a Large Language Model (LLM) for the TSF domain. Again, a follow up paper challenged this by demonstrating that removing the LLM component or replacing it with a basic attention layer in fact yields better performance. One of the challenges in forecasting is the fact that TSF data favors the more recent past, and is sometimes subject to unpredictable events. Based upon these recent insights in TSF, we propose a strong Mixture of Experts (MoE) framework. Our method combines the state-of-the-art (SOTA) models including xLSTM, enhanced Linear, PatchTST, and minGRU, among others. This set of complimentary and diverse models for TSF are integrated in a Transformer based MoE gating network. Our proposed model outperforms all existing TSF models on standard benchmarks, surpassing even the latest approaches based on MoE frameworks.

Our code is available at: <https://github.com/muslehal/EMTSF>

1 Introduction

Time Series Forecasting (TSF) models predict future values based on patterns learned from historical data. TSF is used in many different fields such as weather, health care, traffic, finance, electricity consumption, and market demand, among others. TSF has been a challenging area due to the time dependent nature of data which is often susceptible to factors like seasonality, trend changes, uncommon events, non-stationary nature, and noise issues. While TSF has been an active area of research for decades, it has drawn renewed attention due to the advances in the field of AI. Recent research works have explored AI architectures ranging from simple linear neural networks to enhanced transformer-based architectures, as well as state-space based models. As for the best AI-based architecture for TSF, there have been some interesting antithetical developments in the recently published literature. Before we delve into the details of this perplexity, we briefly describe some of the important research in the field of TSF in a chronological manner.

Some of the early successes in the TSF field have utilized classical statistics and mathematics-based approaches involving moving av-

erage filters, exponential smoothing, and AutoRegressive Integrated Moving Average (ARIMA) processing. By taking into seasonality patterns, techniques such as SARIMA [5] and TBATs [7] provide enhanced prediction as compared to ARIMA. In the last decade, classical machine learning models such as Linear Regression [26], XG-Boost [6], Random Forests and ensemble methods [20] have been explored for the TSF field. These techniques require the data to be transformed into a supervised learning problem by using a sliding window approach. Such models improve performance as they cancel out uncorrelated errors in the averaging process. For larger training data, machine learning approaches usually outperform the classical mathematical techniques of SARIMA and TBATs [7].

The highly successful Convolutional Neural Networks (CNNs) in computer vision were also attempted for TSF. One such example is the Long and Short-term Time-series network (LSTNet) proposed in [13] which used both the CNN and RNN to extract both short and long-term dependency patterns for improving the TSF. Most of the recent research in the TSF domain uses the Transformer architecture that was originally proposed for Natural Language Processing (NLP) [32]. The Transformer uses the attention mechanism to determine the pair-wise similarity in the input sequence to predict the output. It has revolutionized the AI field, leading to development of LLMs such as ChatGPT [1], Llama [31, 8], Gemini [30], DeepSeek [10], among others. Thus, it is natural that these developments be applied to create more effective TSF models.

2 Preliminaries

In a TSF problem, we are given historical time series data $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^{L \times m}$ with m being the number of multivariate time series features and L the number of look-back time steps. The goal is then to train a model that learns to predict the output $\mathbf{y}$, where $\mathbf{y} \in \mathbb{R}^{T \times m}$ is the prediction for the future T time steps. When using a Transformer-based model for TSF, the input data is often divided into a sequence of n patches, i.e., $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, where $\mathbf{x}_i \in \mathbb{R}^{p \times m}$ with $p = \lfloor \frac{L}{n} \rfloor$. These patches are converted to embedding vectors of size $d \times 1$ by a linear transformation according to a fixed dimension d for the transformer model. Position vectors of size $d \times 1$ are added to each embedding vector to maintain relative temporal information.

In the Transformer architecture, three parallel learnt representations of the above transformed data are computed producing $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ matrices, with each $\in \mathbb{R}^{n \times d}$. The Transformer then computes a simple similarity in the form of an inner product on the learnt position encoded embeddings of the sequence of n in-

* Corresponding author Email: muslehal@my.bridgeport.edu

put patches. The pairwise similarity between patches computed as, $\mathbf{A} = \text{softmax}(\mathbf{QK}^T)$ is referred to as the “attention”. If there are n patches being input, referred to as the context, then $\mathbf{A} \in \mathbb{R}^{n \times n}$. Each layer in the Transformer divides the attention calculation into parallel heads by dividing the data along the embedding dimension. The output from a Transformer layer has the same dimensionality as the input, and is obtained by a matrix computation of $(\mathbf{A} \times \mathbf{V}) \in \mathbb{R}^{n \times d}$ where $\mathbf{V} \in \mathbb{R}^{n \times d}$ contains rows of learnt position encoded representations of the input patches. For a Transformer-based TSF implementation, the output is often produced in one step rather than an autoregressive generation of one patch size at a time.

3 Related Work

Some of the important works that use the Transformer architecture for TSF include: Informer [39], Autoformer [36], Fedformer [40], Pyraformer [17], PatchTST [23] and iTransformer [18]. To better adapt the Transformer for TSF, some pre or post processing features have been added to it, e.g., an auto-correlation mechanism is used in Autoformer and series decomposition blocks are included in [36].

Since there is a strong relationship between the time domain and its frequency transformation, Fedformer [40] uses a Fourier enhanced structure. To capture long-range dependencies in data, Pyraformer [17] builds hierarchical representation of the time series by summarizing features at different resolutions, and modeling temporal dependencies across various scales. PatchTST [23] utilizes patching and channel independence to capture dependencies in multivariate time series data. Note that relatively large Transformer models are prone to overfitting in TSF. For example, it has been observed that in the financial time series data, the actual signal is often subtle [38], requiring a much smaller model for better prediction.

Some of the Transformer-based models for TSF have demonstrated better results at the time of their writing, e.g., [39, 36, 17, 40, 18]. However, the LTSF-Linear work in [38] questioned the use of Transformers, reasoning that the permutation-invariant self-attention mechanism may result in temporal information loss. The work in [38] used an extremely simple one layer linear network producing better forecasting results than the previous transformer-based approaches.

One of the most successful Transformer-based TSF implementation has been PatchTST [23]. It segments the time series data into patches, but unlike other models, channel independence is maintained between variates. PatchTST demonstrated significantly superior results as compared to DLinear [38], FEDFormer [40], Autoformer [36] and Informer [39] models. Another important work in adapting the Transformer to the TSF domain is iTransformer [18], which uses the attention mechanism to capture the correlations between different time series (variates) at all historical time points. This allows the model to directly learn how different series influence each other. Since iTransformer does not directly focus on the sequential dependencies within a single time series, this might limit its ability to capture complex, long-range autocorrelations within a series.

The immense success of LLMs in different domains such as NLP, vision, mathematical understanding and reasoning, has raised the question of their viability for the TSF domain. An important work in this context has been recently carried out in [11], referred to as the Time-LLM. Here, the authors reprogram the input time series data with text prototypes before feeding it into the frozen LLM to align the two modalities. Their technique referred to as, Prompt-as-Prefix (PaP), directs the transformation of input TSF patches. The resultant output patches from the LLM are used to obtain the forecasts. The reported results in TIME-LLM outperformed the previous

SOTA forecasting models.

Interestingly, the reasons for the success of TIME-LLM were delineated in a follow-up recent work in [29]. By conducting ablation studies on three LLM-based TSF methods ([41, 16, 11]), the authors demonstrated that removing the LLM component or replacing it with a basic attention layer yields the same performance, and in most cases, in fact improves performance. Further, computationally expensive pre-trained LLMs do not perform better than models trained from scratch. LLM models also do not represent the sequential dependencies in time series well, and do not assist in few-shot settings.

Based on challenges in achieving a single effective model for TSF, more recent research has focused on a Mixture of Experts (MoE) approach, e.g., [22, 28, 24, 27, 14, 33]. The work in [22] argues that even though simple linear transformation based models achieve good results in TSF, due to their inherent simplicity, they are unable to effectively adapt to periodic changes in time series patterns. Thus, they propose an MoE style augmentation for linear-centric models termed Mixture-of-Linear-Experts (MoLE). They train multiple linear-centric models (experts) and a router model that weighs and mixes the expert’s outputs. The results reported in MoLE demonstrated some success over individual linear-centric models of DLinear and RLinear [38], as well as PatchTST [23].

A follow up work in [24] referred to as Mixture of Projection Experts (MoPE) used multiple projection branches instead of a single final projection in a Transformer design to improve the capacity of the network. Similarly, work in [33] uses an MLP-based architecture in an MoE design with Past-Decomposable-Mixing (PDM) and Future-Multipredictor-Mixing (FMM) blocks. This aids in learning disentangled multiscale series in both past and future prediction phases. Another work, termed MOIRAI [14, 27] is an MoE transformer-based model that uses a different patch size for each granularity and learns a mixture of distributions. It also introduces an elegant attention mechanism that respects permutation variance between each variate to capture the temporal dynamics between data points. While MOIRAI is good for multi-source and multi-resolution data, it may not effectively capture long term time series relationships.

Following the LLM philosophy, Time-MoE [28] used a massive data (300 billion time points) approach to train a 2.4 billion parameter LLM designed for TSF. The results surpassed other reported TSF works, however, the complexity of the model and its high training cost are a concern. Further, the Time-300B training dataset which encompasses data from various domains may lead to potential imbalances affecting the model’s generalization capabilities.

Based on the previously mentioned recent research findings and the challenges in effective TSF modeling, we propose a confluent MoE approach combining knowledge from a diverse set of complementary strong experts. We include both linear-based, transformer-based, recently proposed xLSTM based, and minGRU based architectures that we adapt for TSF. We elaborate on these models, and the reasons behind their selection in our MoE framework in the next section. Our contributions can be summarized as:

1. Incorporating strong complimentary experts using linear, transformer, xLSTM and minGRU based approaches that we adapt for TSF in the MoE framework.
2. Development of a confluent MoE architecture where the gating network itself is transformer-based.
3. Detailed insights via empirical results on various datasets to demonstrate the effectiveness of our MoE approach for TSF.
4. Strong results on standard datasets surpassing existing approaches including recent MoE designs.

4 Proposed Method

4.1 EMTSF: Extraordinary Mixture of SOTA Models for Time Series Forecasting

In our proposed EMTSF architecture, different models for TSF are integrated in a Mixture of Experts (MoE) framework as shown in Figure 1. The gating network is designed such that it can control the percentage weight of each expert at each time point in the predicted output. In contrast, in a traditional MoE approach, the gating network is usually a simple linear network that outputs a single coefficient for each expert for all time points. We implement both a Transformer-based gating network as well as a simple linear model for it. We further provide smoothing of the coefficient weights from the gating network by a moving average process, as shown in Figure 1 and Equations 1-2.

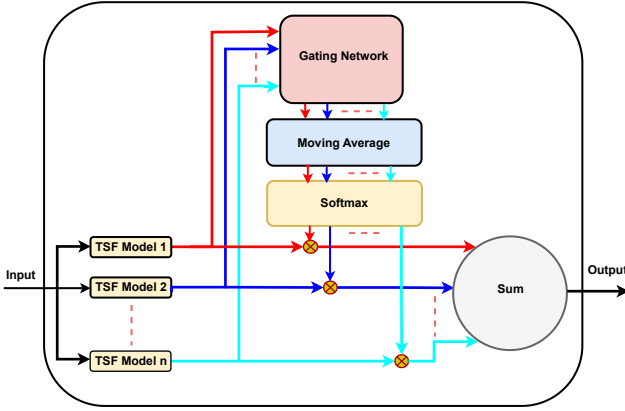


Figure 1. EMTSF: Mixture of Experts Architecture

$$g_i(x) = \text{Softmax}(MA_k(G(\text{cat}(\text{TSF Model}_i(x))))_i \quad (1)$$

where the function G indicates the gating network in MoE (which is itself a transformer network) operating on the sequences of outputs from individual expert TSF models. If the TSF models produce output for T timesteps, then the moving average MA is carried out for k timesteps for each model's gating coefficients where $k \ll T$. This provides smoother combination of different models in the mixture over the range of the predicted timesteps. The final output from the MoE model with n experts is given as:

$$\text{output} = \sum_{i=1}^n g_i(x) \text{TSF Model}_i(x) \quad (2)$$

In the TSF domain, it has been well established that pre-processing the data via series decomposition, batch normalization, and post processing via instance normalization yields better prediction results [22, 38]. Thus, each of the experts (TSF models) in the EMTSF utilizes these processing steps as depicted in Figure 2. The Series

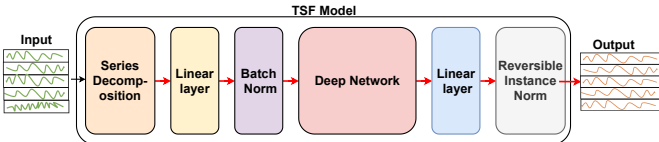


Figure 2. TSF Model with Pre and Post-Processing Blocks

Decomposition block splits the input time series data into two components to capture trend and seasonal information for each series. This approach was proposed in [36] and is formally described as:

For the input sequence with context length of L and m variates, i.e., $\mathbf{x} \in \mathbb{R}^{L \times m}$, learnable moving averages are applied on each feature via 1-D convolutions. The trend and seasonal components are then extracted as:

$$\mathbf{x}_{trend} = \text{AveragePool}(\text{Padding}(\mathbf{x})) \quad (3)$$

$$\mathbf{x}_{seasonal} = \mathbf{x} - \mathbf{x}_{trend} \quad (4)$$

After series decomposition, the data passes through a linear transformation layer to transform it to the dimensionality needed for the deep network for TSF. Batch normalization is further applied to improve stability in learning. The predicted output is transformed back to individual series components via another linear transformation.

A post processing step of Reversible Instance Normalization (RevIN) [12] has proven to improve the TSF performance [22]. The RevIn operates on each channel independently. It applies a learnable transformation to normalize the data during training, such that it can be reversed to its original scale during prediction. The loss functions used in the training of a TSF model are the standard $L2$ (MSE) and $L1$ (MAE) losses.

4.2 Expert TSF Models in EMTSF

Our EMTSF framework is generalizable and any individual TSF model can be integrated in it. For our current implementation, we chose four complementary models that have individually proven to be very effective in the TSF domain. The TSF models selected are: PatchTST [23], Enhanced Linear Model (ELM) [2], xLSTM-Time [3], and a new model for TSF that we develop in this work based on the recent minGRU design [9]. We term this model for TSF as minGRUTime. From a deep learning architecture stand point, one model in EMTSF is based on the Transformer, another is based on a linear network, and the two others are recent enhanced forms of recurrent models. We briefly describe the design of each one of these in the following subsections.

4.2.1 PatchTST Model

PatchTST [23] is one of the most successful Transformer-based models for TSF. Its key design includes segmentation of time series into subseries-level patches, and channel-independence where each component in the multivariate series is treated as a single univariate time series. The embedding and Transformer weights are shared across all the series.

Formally, the i_{th} series for L time steps is treated as a univariate $x_{1:L}^{(i)} = (x_1^{(i)}, \dots, x_L^{(i)})$. Each of these is fed independently to the Transformer after converting to patches. The Transformer then learns to predict the output as $\hat{x} = (\hat{x}_{L+1}^{(i)}, \dots, \hat{x}_{L+T}^{(i)}) \in \mathbb{R}^{1 \times T}$ for the T future steps. For a patch length P and stride S , the patching process generates a sequence of n patches $x_p^{(i)} \in \mathbb{R}^{P \times N}$ where $n = \lfloor \frac{(L-P)}{S} \rfloor + 2$. We chose the PatchTST as one of the TSF models in our EMTSF architecture as the patching design retains local semantic information in the embedding, and the model can effectively attend to the past history due to the attention mechanism of the Transformer.

4.2.2 Enhanced Linear Model

The success of simple linear network based models (e.g., DLinear and NLinear) in [38] resulted in their use in subsequent works for

TSF e.g., [22]. The work in [2] further used dual pipelines to improve the results of DLinear and NLinear. Their model is referred to as Enhanced Linear Model (ELM). One of the pipelines in ELM uses DLinear approach while the other uses NLinear. The advantage of NLinear is that it is able to better handle the distribution shift in the data. It does so by subtracting the last value of the sequence, which it adds it back after the linear layer, and before doing the final prediction.

In ELM, similar to PatchTST, channel independence is maintained. The architecture also utilizes batch normalization and reversible instance normalization [12]. They also used a customized loss function combining the $L1$ and $L2$ losses as:

$$Loss = \alpha \times ||y - \hat{y}||_2 + (1 - \alpha)||y - \hat{y}||_1 \quad (5)$$

The motivation for incorporating ELM with dual pipelines of DLinear and NLinear in our EMTSF architecture is that DLinear is particularly useful for time series with trend and seasonality, while NLinear is effective in handling distribution shifts in TSF.

4.2.3 xLSTMTIME Model

LSTMs (Long Short-Term Memory) networks, which are an improvement over Recurrent Neural Network (RNN) can efficiently handle sequential data such as time series, or NLP. With the emergence of Transformers, their use had declined due to slow and unstable training. In a recent research, the LSTM architecture was revised and significantly improved, termed as xLSTM (Extended LSTM) [4]. xLSTM greatly enhances the traditional LSTM architecture by borrowing some ideas from the Transformer architecture. It also introduces exponential gating for better normalization and stabilization, and the integration of residual block backbones. These improvements give xLSTM scalability and stability to perform competitively with state-of-the-art Transformers [4]. xLSTM has a scalar and matrix variant which are termed sLSTM and mLSTM respectively.

The mLSTM introduces a matrix memory cell along with a covariance update mechanism for key-value pair storage which significantly increases the model's memory capacity. The work in [3] termed as xLSTMTIME utilized both the sLSTM and the mLSTM designs in applying it to the TSF domain. Their implementation added the pre and post-processing stages as shown in Figure 2. sLSTM architecture is employed for smaller datasets and mLSTM for larger datasets with relatively higher number of covariates in TSF.

Since the TSF prediction generally favors more recent past for short term prediction, the exponential gating in xLSTM is naturally suited for this. In addition, the LSTM capability in it helps with the long term forecasting, making xLSTM a very effective architecture for TSF. For these reasons, xLSTMTIME is one of the expert TSF models in our EMTSF architecture.

4.2.4 minGRUTIME Model

Similar to the renewed LSTM design in xLSTM, the work by [9] improves traditional Gated Recurrent Unit (GRU). Their improved version is referred to as minGRU (they also proposed minLSTM). These have fewer parameters and are fully parallelizable during training. It is shown that their performance can rival Transformers. The operation of the minGRU is described by the following equations:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (6)$$

$$z_t = \sigma(\text{Linear}_{d_h}(x_t)) \quad (7)$$

$$\tilde{h}_t = \text{Linear}_{d_h}(x_t) \quad (8)$$

where $\tilde{h}_t$ represents the candidate hidden state, a potential new value for the hidden state, $z_t \in (0, 1)$ determines how much of the past information should be carried forward.

Because of the much simpler design in minGRU, and since TSF domain has shown to be less beneficial from more complex models, we incorporate the minGRU design in our EMTSF architecture. The pre and post processing stages are added to the minGRU as indicated in Figure 2.

Thus overall, we have selected the most promising TSF models based on diverse architectural foundations in our extraordinary mixture of experts (EMTSF). We further use a learnable end-to-end MoE design with a Transformer-based gating network (as indicated in Figure 1). This allows the MoE to extract the best performance by combining the models in a complementary manner. The results in the next section highlight the impressive performance of our EMTSF design over the individual experts including existing state of the art TSF models, as well as recent MoE based TSF designs.

5 Results

We test our EMTSF architecture on different datasets. These include the PEMS traffic dataset (from California Transportation Agencies) which can be used for traffic flow prediction, congestion detection, and travel time estimation. We use the PEMS03, PEMS04, PEMS07 and PEMS08 sets from the above dataset. In addition, we analyze our models on 11 widely used datasets from real-world applications that include the Electricity Transformer Temperature (ETT) series. These are further divided into ETTh1 and ETTh2 (hourly intervals), and ETTm1 and ETTm2 (5-minute intervals). We also perform tests on datasets related to Traffic (hourly), Electricity (hourly), Weather (10-minute intervals), and Influenza-Like Illness (ILI) (weekly). The characteristics of the datasets used in our experiments are detailed in Table 1

Datasets	Timesteps	Features	Granularity
PEMS03	26,209	358	5 min
PEMS04	16,992	307	5 min
PEMS07	28,224	883	5 min
PEMS08	17,856	170	5 min
Weather	52,696	21	10 min
Traffic	17,544	862	1 h
Electricity	26,304	321	1 h
Illness	966	7	1 week
ETTh1/ETTh2	17,420	7	1 h
ETTM1/ETTM2	69,680	7	5 min

Table 1. Characteristics of Time Series Datasets

Table 2 shows comparison of our EMTSF model with other state-of-the-art (SOTA) models for TSF on the popular datasets. The evaluation metrics used are MSE (Mean Squared Error) and MAE (Mean Absolute Error). As it can be seen that in a vast majority of the cases, our EMTSF model outperforms current SOTA designs. The SOTA models compared in Table 2 include: TimeLLM [11], GPT4TS [41], DLinear [38], PatchTST [23], TimesNet [37], FEDFormer [40], Autoformer [36], Stationary [19] and ETSformer [35]. The look back window size in Table 2 is $L=512$ for all datasets except for ILI which uses $L=96$. The predicted length T varies with values of $\{96, 192, 336, 720\}$ and is listed under the Horizon column. For the ILI dataset, the predicted length is different as shown in Table 2. The red color indicates the best result while the blue indicates the second

best for a given category. As can be seen, our EMTS design produces the best MSE and MAE results in majority of the cases. In one category, PatchTST performs the best (Eth2 with a prediction length of 336). In a few cases, TimeLLM has better results. Overall, TimeLLM is second best amongst the models compared in Table 2.

Table 3 shows the comparison of our EMTSF model with TimeMixer++, TimeMixer [33] and iTransformer [18] models. Here also, our EMTSF model outperforms the TimeMixer and iTransformer models. In almost all categories, our model produces best results (lowest MSE and MAE values), followed by TimeMixer++ as the second best model.

Our EMTSF design with four expert models in the mixture uses only 9.66 million parameters. In comparison, Time-LLM [11] is based on a 6.6 billion parameter LLM, while the Time-MOE [28] uses 2.4 billion parameters.

Table 4 shows the comparison of our EMTSF model with Time-MOE [28] and MOIRAI-MOE [27] TSF models. We obtain the best results in most of the cases, with Time-MOE base and large models performing better on the ETTh1 dataset. Overall, as indicated by the comparison results in Tables 2,3 and 4, our EMTSF model outperforms the existing SOTA models for TSF including those that are based on the MoE design such as Time-MOE [28].

5.1 Ablation Study - EMTSF MoE Design

Our EMTSF design initially uses four complementary models in the MoE framework. The models used are: PatchTST, xLSTMTIME, minGRUTime and ELM (Enhanced Linear Model). Even though these models perform well in TSF, do they effectively combine in an MoE framework to produce better results than individual models? To answer this question, we separately trained each of the models in EMTSF and compared their results with the EMTSF end-to-end MoE trained model (architecture described in Figure 1). Table 5 shows the comparisons of our EMTSF MoE with the individual models in it on the popular datasets for TSF. The forecasting is done for prediction targets of $T = \{96, 192, 336, 720\}$ with a look-back window of $L = 512$. This table shows that EMTSF is significantly better than any of the component models (i.e., xLSTMTIME, PatchTST, minGRUTime and ELM). Further, the second best model varies depending upon the dataset and the length of prediction indicating that the contribution from the expert models is data dependent, and all models complement each other in this respect. As can be seen, the mixture of experts is combining the individual experts in a cooperative manner to enhance the overall accuracy in prediction on the various datasets.

Figure 3 shows the graphs for actual versus predicted time series values for the electricity dataset. Our model learns the periodicity and the variations in the data very nicely. To determine, how well the different experts in our EMTSF MoE are contributing to overall generated output, Figure 4 shows the gating weights for each time step in the prediction for the electricity dataset. The Transformer-based EMTSF gating network selects the weight of each expert according to the learnt contribution for that expert. As can be seen, each expert’s weight varies temporally according to the dynamics of the data. Figure 5 shows the prediction for the weather dataset by our model. Figure 6 shows the contribution of each expert for the weather dataset. Similarly, Figures 7 and 8 show the same for the Etm1 dataset.

Figure 9 shows the average weight over the predicted length for each expert assigned by the Gating Network during the output generation for different datasets. All four experts provide a significant contribution to the overall generated output. This further confirms that there is no expert collapse in our MoE design.

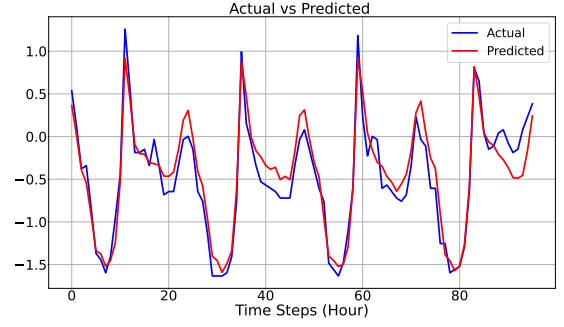


Figure 3. Actual vs. Predicted for the Electricity Dataset

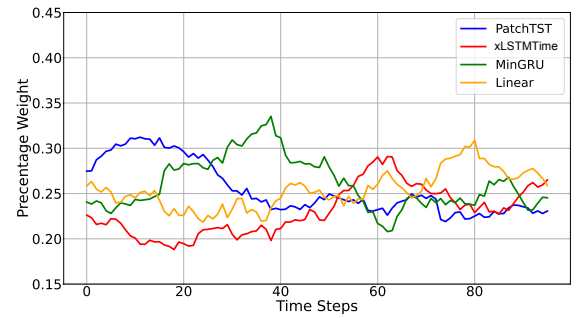


Figure 4. Percentage Gating Weights of different Models in EMTSF for Electricity Dataset

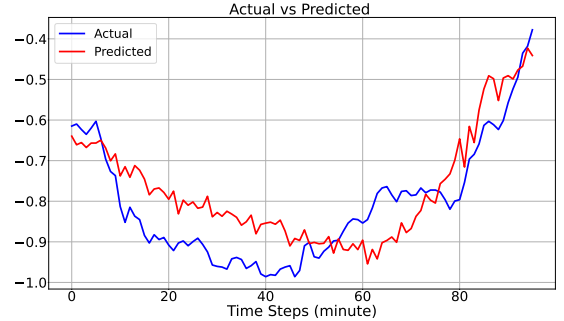


Figure 5. Actual vs. Predicted for the Weather Dataset

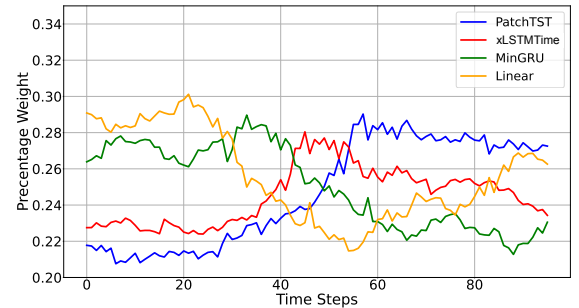


Figure 6. Percentage Gating Weights of different Models in EMTSF for Weather Dataset

Dataset	Horizon	EMTSF(ours)		TIME-LLM		GPT4TS		DLinear		PatchTST		TimesNet		FEDformer		Autoformer		Stationary		ETSformer	
Metric		MSE MAE		MSE MAE		MSE MAE		MSE MAE		MSE MAE		MSE MAE		MSE MAE		MSE MAE		MSE MAE		MSE MAE	
ETTh1	96	0.359	0.384	0.362	0.392	0.376	0.397	0.375	0.399	0.370	0.399	0.384	0.402	0.376	0.419	0.449	0.459	0.513	0.491	0.494	0.479
	192	0.399	0.411	0.398	0.418	0.416	0.418	0.405	0.416	0.413	0.421	0.436	0.429	0.420	0.448	0.500	0.482	0.534	0.504	0.538	0.504
	336	0.418	0.422	0.430	0.427	0.442	0.433	0.439	0.443	0.422	0.436	0.491	0.469	0.459	0.465	0.521	0.496	0.588	0.535	0.574	0.521
	720	0.436	0.454	0.442	0.457	0.477	0.456	0.472	0.490	0.447	0.466	0.521	0.500	0.506	0.507	0.514	0.512	0.643	0.616	0.562	0.535
	Avg	0.403	0.417	0.408	0.423	0.422	0.437	0.413	0.430	0.458	0.450	0.440	0.460	0.496	0.487	0.570	0.537	0.542	0.510	0.491	0.479
ETTh2	96	0.262	0.324	0.268	0.328	0.285	0.342	0.289	0.353	0.274	0.336	0.340	0.374	0.358	0.397	0.346	0.388	0.476	0.458	0.340	0.391
	192	0.328	0.371	0.329	0.375	0.354	0.389	0.383	0.418	0.339	0.379	0.402	0.414	0.429	0.439	0.456	0.452	0.512	0.493	0.430	0.439
	336	0.347	0.387	0.368	0.409	0.373	0.407	0.448	0.465	0.329	0.380	0.452	0.452	0.496	0.487	0.482	0.486	0.552	0.551	0.485	0.479
	720	0.381	0.417	0.372	0.420	0.406	0.441	0.605	0.551	0.379	0.422	0.462	0.468	0.463	0.474	0.515	0.511	0.562	0.560	0.500	0.497
	Avg	0.329	0.374	0.334	0.383	0.381	0.412	0.431	0.446	0.330	0.379	0.414	0.427	0.437	0.449	0.450	0.459	0.526	0.516	0.439	0.452
ETTm1	96	0.271	0.325	0.272	0.334	0.292	0.346	0.299	0.343	0.290	0.342	0.338	0.375	0.379	0.419	0.505	0.475	0.386	0.398	0.375	0.398
	192	0.322	0.351	0.310	0.358	0.332	0.372	0.335	0.365	0.332	0.369	0.374	0.387	0.426	0.441	0.553	0.496	0.459	0.444	0.408	0.410
	336	0.350	0.370	0.352	0.384	0.366	0.394	0.369	0.386	0.366	0.392	0.410	0.411	0.445	0.459	0.621	0.537	0.495	0.464	0.435	0.428
	720	0.414	0.404	0.383	0.411	0.417	0.421	0.425	0.421	0.416	0.420	0.478	0.450	0.543	0.490	0.671	0.561	0.585	0.516	0.499	0.462
	Avg	0.339	0.362	0.329	0.372	0.388	0.403	0.357	0.378	0.351	0.380	0.400	0.406	0.448	0.452	0.588	0.517	0.481	0.456	0.429	0.425
ETTm2	96	0.156	0.240	0.161	0.253	0.173	0.262	0.167	0.269	0.165	0.255	0.187	0.267	0.203	0.287	0.255	0.339	0.192	0.274	0.189	0.280
	192	0.212	0.280	0.219	0.293	0.229	0.301	0.224	0.303	0.220	0.292	0.249	0.309	0.269	0.328	0.281	0.340	0.280	0.339	0.253	0.319
	336	0.263	0.315	0.271	0.329	0.286	0.341	0.281	0.342	0.274	0.329	0.321	0.351	0.325	0.366	0.339	0.372	0.334	0.361	0.314	0.357
	720	0.351	0.371	0.352	0.379	0.378	0.401	0.397	0.421	0.362	0.385	0.408	0.403	0.421	0.415	0.433	0.432	0.417	0.413	0.414	0.413
	Avg	0.245	0.301	0.251	0.313	0.284	0.339	0.267	0.333	0.255	0.315	0.291	0.333	0.305	0.349	0.327	0.371	0.306	0.347	0.293	0.342
Weather	96	0.138	0.177	0.147	0.201	0.162	0.212	0.176	0.237	0.149	0.198	0.172	0.220	0.217	0.296	0.266	0.336	0.173	0.223	0.197	0.281
	192	0.181	0.220	0.189	0.234	0.204	0.248	0.220	0.282	0.194	0.241	0.219	0.261	0.276	0.336	0.307	0.367	0.245	0.285	0.237	0.312
	336	0.230	0.260	0.262	0.279	0.254	0.286	0.265	0.319	0.245	0.282	0.280	0.306	0.339	0.380	0.359	0.395	0.321	0.338	0.298	0.353
	720	0.304	0.315	0.304	0.316	0.326	0.337	0.333	0.362	0.314	0.334	0.365	0.359	0.403	0.428	0.419	0.428	0.414	0.410	0.352	0.386
	Avg	0.213	0.243	0.225	0.257	0.237	0.270	0.248	0.300	0.225	0.264	0.259	0.287	0.309	0.360	0.338	0.382	0.288	0.314	0.271	0.334
Electricity	96	0.126	0.217	0.131	0.224	0.139	0.238	0.140	0.237	0.129	0.222	0.168	0.272	0.193	0.308	0.201	0.317	0.169	0.273	0.187	0.304
	192	0.144	0.234	0.152	0.241	0.153	0.251	0.153	0.249	0.157	0.240	0.184	0.289	0.201	0.315	0.222	0.334	0.182	0.286	0.199	0.315
	336	0.158	0.248	0.160	0.248	0.169	0.266	0.169	0.267	0.163	0.259	0.198	0.300	0.214	0.329	0.231	0.338	0.200	0.304	0.212	0.329
	720	0.190	0.277	0.192	0.298	0.206	0.297	0.203	0.301	0.197	0.290	0.220	0.320	0.246	0.355	0.254	0.361	0.222	0.321	0.233	0.345
	Avg	0.154	0.244	0.158	0.252	0.167	0.263	0.166	0.263	0.161	0.252	0.192	0.295	0.214	0.327	0.227	0.338	0.193	0.296	0.208	0.323
Traffic	96	0.343	0.225	0.362	0.248	0.388	0.282	0.410	0.282	0.360	0.249	0.593	0.321	0.587	0.366	0.613	0.388	0.612	0.338	0.607	0.392
	192	0.369	0.238	0.374	0.247	0.407	0.290	0.423	0.287	0.379	0.256	0.617	0.336	0.604	0.373	0.616	0.382	0.613	0.340	0.621	0.399
	336	0.382	0.242	0.385	0.271	0.412	0.294	0.436	0.296	0.392	0.264	0.629	0.336	0.621	0.383	0.622	0.337	0.618	0.328	0.622	0.396
	720	0.424	0.270	0.430	0.288	0.450	0.312	0.466	0.315	0.432	0.286	0.640	0.350	0.626	0.382	0.660	0.408	0.653	0.355	0.632	0.396
	Avg	0.379	0.243	0.388	0.264	0.414	0.294	0.433	0.295	0.390	0.263	0.620	0.336	0.610	0.376	0.628	0.379	0.624	0.340	0.621	0.396
ILI	24	1.617	0.732	1.285	0.727	2.063	0.881	2.215	1.081	1.319	0.754	2.317	0.934	3.228	1.260	3.483	1.287	2.944	0.945	2.527	1.020
	36	1.586	0.728	1.404	0.814	1.868	0.892	1.963	0.963	1.430	0.834	1.972	0.920	2.679	1.080	3.103	1.148	1.825	0.848	2.615	1.007
	48	1.587	0.753	1.523	0.807	1.790	0.884	2.130	1.024	1.553	0.815	2.238	0.940	2.622	1.078	2.669	1.085	2.010	0.900	2.359	0.972
	60	1.560	0.768	1.531	0.854	1.979	0.957	2.368	1.096	1.470	0.788	2.027	0.928	2.857	1.157	2.770	1.125	2.178	0.963	2.487	1.016
	Avg	1.587	0.745	1.435	0.801	1.925	0.903	2.169	1.041	1.443	0.797	2.139	0.931	2.847	1.144	3.006	1.161	2.077	0.914	2.497	1.004

Table 2. Performance Comparison of our EMTSF model with other SOTA Models on popular TSF Datasets.

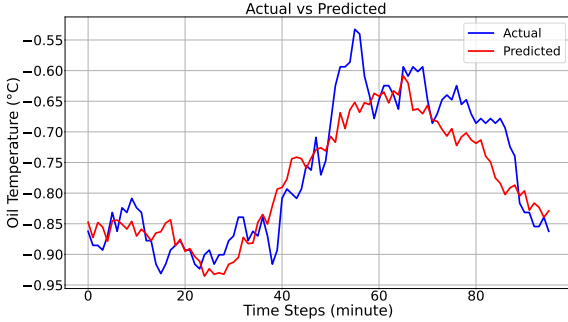


Figure 7. Actual vs. Predicted for the ETTm1 Dataset

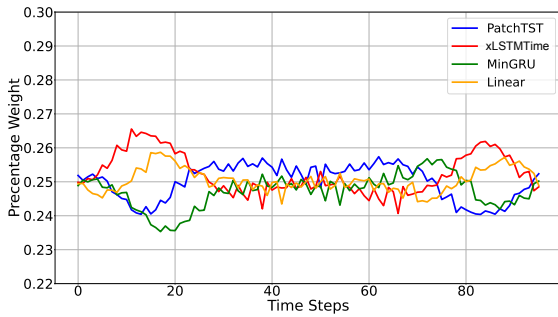


Figure 8. Percentage Gating Weights of different Models in EMTSF for ETTm1 Dataset

Dataset	Horizon	EMTSF (Ours)		TimeMixer++		TimeMixer (2024b)		iTransformer (2024)	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.359	0.384	0.361	0.403	0.375	0.400	0.386	0.405
	192	0.399	0.411	0.416	0.441	0.429	0.421	0.441	0.512
	336	0.418	0.422	0.430	0.434	0.484	0.458	0.487	0.458
	720	0.436	0.454	0.467	0.451	0.498	0.482	0.503	0.491
	Avg	0.403	0.417	0.419	0.432	0.447	0.440	0.454	0.447
ETTh2	96	0.262	0.324	0.276	0.328	0.289	0.341	0.297	0.349
	192	0.328	0.371	0.342	0.379	0.372	0.392	0.380	0.400
	336	0.347	0.387	0.346	0.398	0.386	0.414	0.428	0.432
	720	0.381	0.417	0.392	0.415	0.412	0.434	0.427	0.445
	Avg	0.329	0.374	0.339	0.380	0.364	0.395	0.383	0.407
ETTm1	96	0.271	0.325	0.310	0.334	0.320	0.357	0.334	0.368
	192	0.322	0.351	0.348	0.362	0.361	0.381	0.390	0.393
	336	0.350	0.370	0.376	0.391	0.390	0.404	0.426	0.420
	720	0.414	0.404	0.440	0.423	0.454	0.441	0.491	0.459
	Avg	0.339	0.362	0.369	0.378	0.381	0.395	0.407	0.410
ETTm2	96	0.156	0.240	0.170	0.245	0.175	0.258	0.180	0.264
	192	0.212	0.280	0.229	0.291	0.237	0.299	0.250	0.309
	336	0.263	0.315	0.303	0.343	0.298	0.340	0.311	0.348
	720	0.351	0.371	0.373	0.399	0.391	0.396	0.412	0.407
	Avg	0.245	0.301	0.269	0.320	0.275	0.323	0.288	0.332
Weather	96	0.138	0.177	0.155	0.205	0.163	0.209	0.174	0.214
	192	0.181	0.220	0.201	0.245	0.208	0.250	0.221	0.254
	336	0.230	0.260	0.237	0.263	0.251	0.287	0.278	0.296
	720	0.304	0.315	0.312	0.334	0.339	0.341	0.358	0.347
	Avg	0.213	0.243	0.226	0.262	0.240	0.271	0.258	0.278
Electricity	96	0.126	0.217	0.135	0.222	0.153	0.247	0.148	0.240
	192	0.144	0.234	0.147	0.235	0.166	0.256	0.162	0.253
	336	0.158	0.248	0.164	0.245	0.185	0.277	0.178	0.269
	720	0.190	0.277	0.212	0.310	0.225	0.310	0.225	0.317
	Avg	0.154	0.244	0.165	0.253	0.182	0.272	0.178	0.270
Traffic	96	0.343	0.225	0.392	0.253	0.462	0.285	0.395	0.268
	192	0.369	0.238	0.402	0.258	0.473	0.296	0.417	0.276
	336	0.382	0.242	0.428	0.263	0.498	0.296	0.433	0.283
	720	0.424	0.270	0.441	0.282	0.506	0.313	0.467	0.302
	Avg	0.379	0.243	0.416	0.264	0.484	0.297	0.428	0.282

Dataset	Horizon	EMTSF (Ours)		TIME-MOE _{base}		TIME-MOE _{large}		TIME-MOE _{extra}		MoIRAI _{base}		MoIRAI _{large}		MoIRAI _{extra}	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.359	0.384	0.357	0.381	0.350	0.382	0.349	0.379	0.401	0.402	0.376	0.392	0.381	0.388
	192	0.399	0.411	0.384	0.404	0.388	0.412	0.395	0.413	0.435	0.421	0.412	0.413	0.431	0.400
	336	0.418	0.422	0.411	0.434	0.411	0.430	0.447	0.453	0.453	0.453	0.443	0.445	0.495	0.430
	720	0.436	0.454	0.449	0.477	0.427	0.455	0.457	0.462	0.439	0.454	0.487	0.444	0.611	0.456
	Avg	0.403	0.417	0.404	0.424	0.394	0.419	0.412	0.426	0.428	0.427	0.417	0.419	0.480	0.419
ETTh2	96	0.262	0.354	0.305	0.359	0.302	0.354	0.292	0.352	0.297	0.336	0.294	0.330	0.296	0.330
	192	0.328	0.371	0.351	0.386	0.364	0.386	0.347	0.379	0.362	0.375	0.363	0.371	0.370	0.371
	336	0.347	0.387	0.391	0.418	0.417	0.425	0.366	0.419	0.370	0.393	0.376	0.390	0.393	0.384
	720	0.381	0.417	0.419	0.454	0.537	0.496	0.439	0.447	0.411	0.426	0.416	0.433	0.423	0.418
	Avg	0.329	0.374	0.366	0.404	0.405	0.415	0.371	0.399	0.361	0.384	0.362	0.382	0.367	0.377
ETTm1	96	0.271	0.325	0.338	0.368	0.309	0.357	0.281	0.341	0.418	0.392	0.363	0.356	0.380	0.361
	192	0.322	0.351	0.353	0.388	0.346	0.381	0.305	0.358	0.431	0.405	0.388	0.375	0.412	0.388
	336	0.350	0.370	0.381	0.413	0.373	0.408	0.360	0.395	0.460	0.418	0.392	0.436	0.434	0.404
	720	0.414	0.404	0.504	0.493	0.475	0.477	0.469	0.472	0.462	0.432	0.460	0.418	0.462	0.420
	Avg	0.339	0.362	0.394	0.415	0.376	0.405	0.356	0.391	0.436	0.410	0.406	0.385	0.422	0.391
ETTm2	96	0.156	0.240	0.201	0.291	0.197	0.286	0.198	0.288	0.214	0.288	0.205	0.273	0.211	0.274
	192	0.212	0.280	0.258	0.334	0.250	0.322	0.235	0.312	0.284	0.332	0.275	0.316	0.281	0.318
	336	0.263	0.315	0.324	0.373	0.337	0.375	0.293	0.448	0.331	0.362	0.329	0.350	0.341	0.355
	720	0.331	0.371	0.488	0.464	0.480	0.461	0.427	0.428	0.402	0.408	0.385	0.423	0.485	0.426
	Avg	0.245	0.301	0.317	0.365	0.316	0.361	0.288	0.344	0.307	0.347	0.337	0.371	0.329	0.343
Weather	96	0.138	0.177	0.160	0.214	0.159	0.213	0.157	0.211	0.198	0.222	0.220	0.217	0.199	0.211
	192	0.181	0.220	0.210	0.260	0.215	0.266	0.208	0.256	0.247	0.265	0.271	0.259	0.246	0.251
	336	0.230	0.260	0.274	0.309	0.291	0.322	0.255	0.290	0.283	0.303	0.286	0.297	0.274	0.291
	720	0.304	0.315	0.418	0.405	0.415	0.400	0.405	0.397	0.373	0.354	0.373	0.354	0.337	0.340
	Avg	0.213	0.243	0.265	0.297	0.270	0.300	0.256	0.288	0.275	0.286	0.273	0.281	0.264	0.273

Table 4. Comparison of EMTSF (ours) with TIME-MOE and MoIRAI-MOE using MSE and MAE (Red = Best, Blue = 2nd Best)

Models	Horizon	EMTSF (MoE)		xLSTMTime		PatchTST		minGRUTime		ELM	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.138	0.177	0.147	0.192	0.143	0.180	0.143	0.182	0.160	0.187
	192	0.182	0.220	0.194	0.237	0.187	0.224	0.187	0.228	0.185	0.222
	336	0.232	0.260	0.242	0.275	0.240	0.267	0.236	0.266	0.242	0.264
	720	0.305	0.315	0.315	0.329	0.312	0.318	0.311	0.320	0.311	0.319
	Avg	0.213	0.243	0.225	0.259	0.217	0.241	0.217	0.241	0.217	0.241
Traffic	96	0.343	0.225	0.341	0.229	0.376	0.241	0.396	0.265	0.412	0.263
	192	0.369	0.238	0.365	0.242	0.387	0.241	0.415	0.269	0.425	0.266
	336	0.382	0.242	0.382	0.245	0.398	0.246	0.424	0.275	0.431	0.270
	720	0.424	0.270	0.425	0.270	0.437	0.272	0.458	0.300	0.465	0.289
	Avg	0.356	0.241	0.356	0.241	0.376	0.241	0.415	0.269	0.425	0.266
Electricity	96	0.126	0.177	0.130	0.173	0.127	0.173	0.127	0.173	0.127	0.173
	192	0.144	0.234	0.149	0.241	0.144	0.233	0.149	0.240	0.147	0.236
	336	0.158	0.248	0.165	0.258	0.160	0.250	0.166	0.257	0.163	0.253
	720	0.190	0.277	0.191	0.279	0.193	0.279	0.205	0.291	0.204	0.288
	Avg	0.157	0.213	0.157	0.213	0.157	0.213	0.157	0.213	0.157	0.213
Illness	24	1.617	0.732	1.639	0.733	1.670	0.784	1.665	0.750	1.939	0.808
	36	1.586	0.728	1.486	0.711	1.709	0.789	1.648	0.758	1.838	0.806
	48	1.587	0.753	1.461	0.742	1.748	0.800	1.599	0.762	1.783	0.817
	60	1.560	0.768	1.617	0.798	1.608	0.790	1.543	0.773	1.760	0.836
	Avg	0.359	0.384	0.444	0.455	0.372	0.397	0.362	0.387	0.372	0.389
ETTh1	96	0.359	0.411	0.463	0.462	0.429	0.434	0.402	0.414	0.412	0.416
	192	0.399	0.411	0.463	0.462	0.429	0.434	0.402	0.414	0.412	0.416
	336	0.418	0.422	0.487	0.481	0.438	0.438	0.436	0.429	0.448	0.433
	720	0.436	0.454	0.526	0.523	0.460	0.468	0.446	0.455	0.476	0.461
	Avg	0.362	0.374	0.421	0.427	0.376	0.376	0.372	0.372	0.372	0.389
ETTh2	96	0.262	0.324	0.281	0.342	0.279	0.336	0.272	0.331	0.269	0.331
	192	0.328	0.371	0.357	0.387	0.352	0.385	0.341	0.377	0.331	0.372
	336	0.347	0.387	0.397	0.421	0.367	0.400	0.363	0.400	0.358	0.399
	720	0.381	0.417	0.410	0.438	0.399	0.430	0.403	0.432	0.438	0.424
	Avg	0.329	0.374	0.366	0.404	0.405	0.415	0.371	0.399	0.361	0.384
ETTm1	96	0.271	0.325	0.338	0.368	0.309	0.357	0.281	0.341	0.418	0.392
	192	0.322	0.351	0.353	0.388	0.346	0.381	0.305	0.358	0.431	0.405
	336	0.350	0.370	0.381	0.413	0.373	0.408	0.360	0.395	0.460	0.418
	720	0.414	0.404	0.504	0.493	0.475	0.477	0.469	0.472	0.462	0.432
	Avg	0.339	0.362	0.394	0.415	0.376	0.405	0.356	0.391	0.436	0.410
ETTm2	96	0.156	0.240	0.201	0.291	0.197	0.286	0.198	0.288	0.214	0.288
	192	0.212	0.280	0.258	0.334	0.250	0.322	0.235	0.312	0.284	0.332
	336	0.263	0.315	0.324	0.373	0.337	0.375	0.293	0.448	0.331	0.362
	720	0.331	0.371	0.488	0.464	0.480	0.461	0.427	0.428	0.402	0.408
	Avg	0.245	0.301	0.317	0.365	0.316	0.361	0.288	0.344	0.307	0.347

Table 5. Performance Comparison of EMTSF MoE Design with Component Models on popular TSF Datasets

Some recent TSF models such as WPMixer [21], CycleNet [15], and TimeXer [34] have proposed interesting TSF ideas. WPMixer is a novel MLP-based model that leverages the benefits of patching, multi-resolution wavelet decomposition, and mixing for TSF. CycleNet uses the "Residual Cycle Forecasting" technique, which utilizes learnable recurrent cycles to model the inherent periodic patterns within sequences. TimeXer enhances the Transformer's capability to reconcile endogenous and exogenous information, where patch-wise self-attention and variate-wise cross-attention are used simultaneously. As can be seen from Table 7, WPMixer is competitive with our EMTSF model for the ETT datasets, but lags in other datasets. This could be attributed to the Wavelet backbone being able to capture the hourly or five minute patterns of the transformer temperature in the ETTh/m datasets. WPMixer can be a good candidate model as one of the experts in our EMTSF framework.

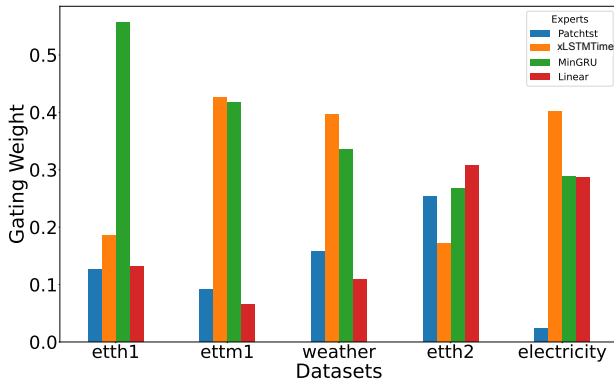


Figure 9. Percentage Gating Weights for different Models in EMTSF for different Datasets

Models	Horizon	EMTSF (ours)		xLSTMTime		PatchTST		minGRUTime		ELM	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
PEMS03	12	0.061	0.162	0.064	0.166	0.075	0.181	0.071	0.176	0.128	0.239
	24	0.083	0.190	0.084	0.189	0.107	0.216	0.103	0.212	0.315	0.414
	48	0.127	0.236	0.132	0.235	0.177	0.276	0.168	0.275	0.714	0.631
	96	0.227	0.326	0.257	0.352	0.346	0.398	0.280	0.361	1.703	1.035
	Avg % Improvement	7.46%	2.97%								
PEMS04	12	0.072	0.171	0.072	0.173	0.091	0.195	0.093	0.193	0.133	0.247
	24	0.089	0.193	0.093	0.194	0.140	0.245	0.124	0.232	0.247	0.338
	48	0.114	0.222	0.117	0.225	0.187	0.283	0.214	0.313	0.548	0.519
	96	0.154	0.260	0.160	0.263	0.278	0.352	0.284	0.370	1.062	0.766
	Avg % Improvement	2.73%	1.26%								
PEMS07	12	0.057	0.146	0.058	0.147	0.070	0.166	0.278	0.352	0.160	0.263
	24	0.072	0.166	0.072	0.167	0.111	0.211	0.105	0.211	0.232	0.323
	48	0.101	0.196	0.101	0.196	0.201	0.291	0.220	0.321	0.543	0.519
	96	0.130	0.226	0.130	0.227	0.299	0.337	0.254	0.346	1.066	0.748
	Avg % Improvement	0.28%	0.54%								
PEMS08	12	0.070	0.166	0.072	0.169	0.087	0.186	0.082	0.184	0.295	0.406
	24	0.091	0.188	0.097	0.193	0.124	0.222	0.121	0.222	0.233	0.322
	48	0.140	0.228	0.146	0.232	0.210	0.291	0.204	0.294	0.529	0.507
	96	0.222	0.279	0.231	0.284	0.376	0.393	0.351	0.392	1.109	0.754
	Avg % Improvement	4.78%	1.83%								

References

- [1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] M. Alharthi and A. Mahmood. Enhanced linear and vision transformer-based architectures for time series forecasting. *Big Data and Cognitive Computing*, 8(5):48, 2024.
- [3] M. Alharthi and A. Mahmood. xlstmtime: Long-term time series forecasting with xlstm. *AI*, 5(3):1482–1495, 2024.
- [4] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.
- [5] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.
- [6] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [7] A. M. De Livera, R. J. Hyndman, and R. D. Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.
- [8] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [9] L. Feng, F. Tung, M. O. Ahmed, Y. Bengio, and H. Hajimirsadeh. Were rns all we needed? *arXiv preprint arXiv:2410.01201*, 2024.
- [10] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [11] M. Jin, S. Wang, L. Ma, Z. Chu, J. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-f. Li, S. Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *Proceedings of the International Conference on Learning Representations*, 2024.
- [12] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *Proceedings of the International Conference on Learning Representations*, 2021.
- [13] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 95–104, 2018.
- [14] Y. Liang, D. Sahoo, J. Zhou, J. Gao, N. Botten, S. Savarese, and C. Xiong. MOIRAI: Towards a universal model for time series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12938–12961. PMLR, 2024.
- [15] S. Lin, W. Lin, X. Hu, W. Wu, R. Mo, and H. Zhong. Cyclenet: Enhancing time series forecasting through modeling periodic patterns. *Advances in Neural Information Processing Systems*, 37:106315–106345, 2024.
- [16] P. Liu, H. Guo, T. Dai, N. Li, J. Bao, X. Ren, Y. Jiang, and S. T. Xia. Calf: Aligning llms for time series forecasting via cross-modal finetuning. *arXiv preprint arXiv:2403.07300*, 2024.
- [17] 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 *Proceedings of the International Conference on Learning Representations*, 2022.
- [18] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*.
- [19] 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*, 35:9881–9893, 2022.
- [20] R. P. Masini, M. C. Medeiros, and E. F. Mendes. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1):76–111, 2023.
- [21] M. M. N. Murad, M. Aktukmak, and Y. Yilmaz. Wpmixer: Efficient multi-resolution mixing for long-term time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19581–19588, 2025.
- [22] R. Ni, Z. Lin, S. Wang, and G. Fanti. Mixture-of-linear-experts for long-term time series forecasting. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (PMLR)*, pages 4672–4680, 2024.
- [23] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *Proceedings of the International Conference on Learning Representations*, 2023.
- [24] H. Niu, G. Habault, D. Cao, Y. Zhang, R. Legaspi, H. Q. Ung, J. Enouen, S. Wada, C. Ono, A. Minamikawa, et al. Mixture of projection experts for multivariate long-term time series forecasting. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1798–1803. IEEE, 2024.
- [25] J. Puigcerver, C. R. Ruiz, B. Mustafa, and N. Houlsby. From sparse to soft mixtures of experts. In *The Twelfth International Conference on Learning Representations*.
- [26] G. Ristanoski, W. Liu, and J. Bailey. Time series forecasting using distribution enhanced linear regression. In *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I 17*, pages 484–495. Springer, 2013.
- [27] D. Sahoo, Y. Liang, S. Savarese, J. Liu, G. Woo, C. Xiong, R. Zimmermann, and X. Liu. Moirai-moe: Empowering time series foundation models with sparse mixture of experts, 2024.
- [28] X. Shi, S. Wang, Y. Nie, D. Li, Z. Ye, Q. Wen, and M. Jin. Time-moe: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations (ICLR)*, 2025.
- [29] M. Tan, M. A. Merrill, V. Gupta, T. Althoff, and T. Hartvigsen. Are language models actually useful for time series forecasting? In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, 2024.
- [30] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [31] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [32] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [33] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, and L. Ma. Timemixer: Decomposable multiscale mixing for time series forecasting. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.
- [34] Y. Wang, H. Wu, J. Dong, G. Qin, H. Zhang, Y. Liu, Y. Qiu, J. Wang, and M. Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. *Advances in Neural Information Processing Systems*, 37:469–498, 2024.
- [35] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arXiv:2202.01381*, 2022.
- [36] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2021.
- [37] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.
- [38] 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*, volume 37, pages 11121–11128, 2023.
- [39] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.
- [40] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the International Conference on Machine Learning (PMLR)*, pages 27268–27286, 2022.
- [41] T. Zhou, P. Niu, L. Sun, R. Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in Neural Information Processing Systems*, 36:43322–43355, 2023.