FAST TRAINING OF MIXTURE-OF-EXPERTS FOR TIME SERIES FORECASTING VIA EXPERT LOSS INTEGRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a novel Mixture-of-Experts framework, referred to as Expert Loss Integration MoE (EliMoE), for time series forecasting. EliMoE addresses the optimization problem arising from small gating weights by incorporating expert-specific losses, which provide each expert with a direct learning signal independent of the gate-assigned weight. Specifically, the overall objective comprises the base forecasting loss and expert-specific losses, allowing individual expert prediction errors to directly influence parameter updates alongside the aggregate forecasting error. The framework also encourages different experts to learn from different temporal segments of the data. The proposed framework is further combined with a partial online learning strategy that enables efficient incremental updates of model parameters. By integrating expert-level loss information with partial online learning, EliMoE improves forecasting performance while retaining computational efficiency. Empirical results across five benchmark datasets from diverse application domains with different sampling frequencies show that the proposed approach generally outperforms state-of-the-art supervised neural forecasting models, including Transformer-based architectures such as PatchTST, as well as zero-shot time-series foundation models such as TimeMoE. Furthermore, ablation studies confirm the effectiveness of the expert-specific loss integration strategy, highlighting its contribution to enhancing predictive performance.

1 INTRODUCTION

Time series forecasting is a widely used technique for predicting future outcomes, with broad applications across economics, finance, energy systems, and many other domains. Accurate forecasts are crucial for ensuring efficient capacity scheduling, supporting strategic decision-making, and enabling effective risk management. Even small improvements in predictive accuracy can yield substantial economic value Hyndman & Athanasopoulos (2021). However, constructing a precise forecasting model is not easily attainable in a world full of uncertainties, as time series data often exhibit seasonality, trends, structural breaks, and other complex patterns. Another important challenge in time series forecasting, besides forecast accuracy, is the computational efficiency of the model, since decisions often need to be made within strict time constraints.

Classical statistical methods such as ARIMA Box & Jenkins (1990) and exponential smoothing Holt (2004) are good benchmarks, particularly for short, well-behaved series. These methods are computationally efficient and often provide reliable forecasting performance. However, they typically rely on restrictive linearity or stationarity assumptions and tend to perform poorly when those assumptions are violated. Modern neural architectures such as recurrent neural networks (RNNs), convolutional models like WaveNet Borovykh et al. (2017), and attention-based Transformers Vaswani et al. (2017); Zhou et al. (2021) have significantly expanded model capacity and achieved state-of-the-art performance on several forecasting benchmarks. However, these models typically require substantial computational resources and long training times for both model fitting and forecast generation.

Mixture-of-Experts (MoE) architectures could provide a potential solution to the challenges of forecast accuracy and computational efficiency by training multiple expert networks alongside a learned gating mechanism that routes inputs to the most appropriate experts. In practice, however, expert

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

specialization is not guaranteed. When training is driven solely by a global forecasting loss, the gradient signal received by each expert is filtered through the gating weights, which may limit the emergence of distinct expert behaviors.

We introduce a new MoE framework that jointly addresses the challenges of forecasting accuracy and computational efficiency. We refer to this approach as Expert Loss Integration Mixture-of-Experts (EliMoE). We augment the conventional global forecasting loss with expert-specific loss terms. This loss formulation explicitly addresses the weak-gradient problem caused by very small gating weights while encouraging experts to learn from different portions of the time series. We further integrate this objective with a partial online learning strategy, in which the gating network and a subset of expert parameters are updated incrementally as new observations arrive, while shared representational components are refreshed less frequently. This design substantially reduces computational cost while maintaining high forecasting accuracy.

The contributions of this work are as follows. First, we formalize a training objective that combines a global forecasting loss with expert-specific losses, and demonstrate that the proposed objective improves overall forecasting accuracy. Additionally, we apply masking to selected experts to encourage each network to focus on distinct patterns within the data. Second, we combine MoE with a partial online learning scheme that reduces computational cost without sacrificing predictive performance. Third, we evaluate the proposed framework across diverse datasets at multiple sampling frequencies and compare it with competitive neural baselines. Finally, ablation studies isolate the contribution of the expert-specific loss component and confirm its effectiveness in improving forecasting accuracy.

The remainder of the paper is organized as follows. Section 2 reviews related work on time series forecasting and mixture-of-experts models. Section 3 describes the proposed methodology. Section 4 presents the experimental setup and results, and Section 5 concludes.

2 Related work

The Mixture of Experts paradigm was introduced by Jacobs et al. (1991) as a framework in which specialized experts model different regions of the input space, while a gating network determines which expert to select. Jordan & Jacobs (1994) later extended this into hierarchical MoE and proposed an expectation–maximization (EM) training algorithm.

Interest in MoE was revived by Shazeer et al. (2017), who introduced the sparsely-gated MoE layer. By activating only the top-k experts for each input, they showed that model capacity could scale dramatically without proportional computational cost. They also identified two major routing issues: expert collapse, where only a few experts dominate, and load imbalance, where workloads are unevenly distributed. To address this, they proposed auxiliary load-balancing and importance regularization losses.

Later work extended MoE to large-scale Transformer systems. GShard Lepikhin et al. (2021) enabled distributed expert sharding across accelerators, while Switch Transformer Fedus et al. (2022) simplified routing to top-1 expert selection, reducing communication overhead while maintaining strong performance.

A recurring challenge in MoE research is maintaining robust expert specialization. Under task loss alone, rarely selected experts receive weak gradients, while heavily used experts risk becoming overly general. Prior work has therefore focused mainly on routing behavior. Load-balancing losses Shazeer et al. (2017); Fedus et al. (2022), stochastic routing methods such as THOR Zuo et al. (2022), BASE Layers Lewis et al. (2021), and expert-choice routing Zhou et al. (2022) all improve routing stability and balanced utilization. However, these approaches remain largely routing-focused: they regulate where tokens are routed, but do not directly optimize the predictive quality or specialization of individual experts.

Recent work has increasingly applied MoE architectures to time series forecasting. Mixture-of-Linear-Experts (MoLE) Ni et al. (2024) extends linear forecasting models such as DLinear Zeng et al. (2023), RLinear, and RMLP Li et al. (2023) with multiple linear experts and a timestamp-conditioned router, enabling experts to specialize in distinct seasonal or periodic behaviors. LeMoLE Zhang et al. (2024) further expands this framework by incorporating LLM-derived features for multimodal forecasting.

At larger scales, TimeMoE Shi et al. (2025) introduces a sparse MoE-based decoder-only foundation model for time series forecasting. Moirai-MoE Liu et al. (2024) similarly integrates sparse expert layers into a unified time series foundation model. Together, these studies demonstrate that MoE provides an effective inductive bias for time series forecasting. However, like prior MoE research, they primarily rely on global forecasting objectives and router-side auxiliary losses, without directly supervising expert-level prediction quality.

Beyond architectural design, recent work has started to ask whether classical MoE training strategies transfer to forecasting. Yemets et al. (2025) systematically compare three families of load-balancing strategies for MoE-based time-series forecasting and find that classical auxiliary losses imported from language modeling can actively degrade forecasting accuracy. This motivates the exploration of alternative forms of expert-level supervision, which our expert-specific loss formulation is designed to address.

3 Model Design

We propose a multivariate neural MoE framework which incorporates $K+1$ experts. The first expert is a linear model that directly connects the input to its output, denoted by $Y^{\mathrm{Lin}}(\mathbf{x})$. Each of the remaining K experts is a multilayer perceptron (MLP) with one hidden layer, denoted as $Y^{\mathrm{MLP},k}(\mathbf{x})$. The architecture of each MLP may vary, particularly with respect to the number of neurons in the hidden layer. The model is expressed as follows:

$$Y = w_0(\mathbf{x})\, Y^{\mathrm{Lin}}(\mathbf{x}) + \sum_{k=1}^{K} w_k(\mathbf{x})\, Y^{\mathrm{MLP},k}(\mathbf{x}) + \xi,$$

$$Y^{\mathrm{Lin}}(\mathbf{x}) = \mathbf{x}^{\top}\boldsymbol{\beta}, \qquad Y^{\mathrm{MLP},k}(\mathbf{x}) = \mathbf{W}_k^{(2)}\,\sigma\Big(\mathbf{W}_k^{(1)}\,\mathbf{x} + \mathbf{b}_k^{(1)}\Big) + \mathbf{b}_k^{(2)}.$$

where $\mathbf{x} \in \mathbb{R}^m$ is the input vector with m regressors. $\boldsymbol{\beta}$ is the vector of weights for the skip connection. $\mathbf{W}_k^{(1)}, \mathbf{W}_k^{(2)}, \mathbf{b}_k^{(1)}, \mathbf{b}_k^{(2)}$ are the trainable weight matrices and bias vectors, respectively, for k = 1,...,K. σ is the activation function for the hidden layer. $w_j(\mathbf{x})$ denotes the weight assigned to each expert by the gating mechanism and expressed as

$$w_j(\mathbf{x}) = \frac{\exp(g_j(\mathbf{x}))}{\sum_{\ell=0}^{K}\exp(g_\ell(\mathbf{x}))}, \qquad g_j(\mathbf{x}) = \sigma\big(\mathbf{a}_j^{\top}\mathbf{x} + c_j\big), \qquad j = 0,\ldots,K,$$

where $\mathbf{a}_j$ denotes the vector of weights connecting the input features to the gating neuron associated with expert j and c_j is its bias term. Figure 1 illustrates the architecture of EliMoE. The blue connections correspond to the expert networks. The orange gating weights are generated from neurons directly connected to the input, followed by a LeakyReLU activation and a softmax function to obtain the normalized expert weights $w_j(\mathbf{x})$.

To estimate the parameters of EliMoE, we introduce a novel objective function that incorporates the individual losses of each expert, as defined below:

$$\mathcal{L}(\boldsymbol{\theta}) = \gamma\,\mathcal{L}_{\mathrm{base}} + (1-\gamma)\,\mathcal{L}_{\mathrm{aux}}, \tag{1}$$

$$\mathcal{L}_{\mathrm{base}} = \frac{1}{N}\sum_{n=1}^{N}\left|Y_n - \widehat{Y}_n\right|,$$

$$\mathcal{L}_{\mathrm{aux}} = \frac{1}{K+1}\left(\frac{1}{N}\sum_{n=1}^{N}\left|Y_n - \widehat{Y}_n^{\mathrm{Lin}}\right| + \sum_{k=1}^{K}\frac{1}{N-\lfloor \pi_k N\rfloor}\sum_{n=1}^{N}\mathbb{I}(n > \lfloor \pi_k N\rfloor)\left|Y_n - \widehat{Y}_n^{\mathrm{MLP},k}\right|\right),$$

$$\text{with } \pi_k = \frac{k-1}{K}, \quad \gamma \in [0,1] \quad \text{and } k = 1,\ldots,K.$$

The objective function in equation 1 consists of two components. The term $\mathcal{L}_{\mathrm{base}}$ represents the overall prediction error of the MoE model, measured using the mean absolute error. This term

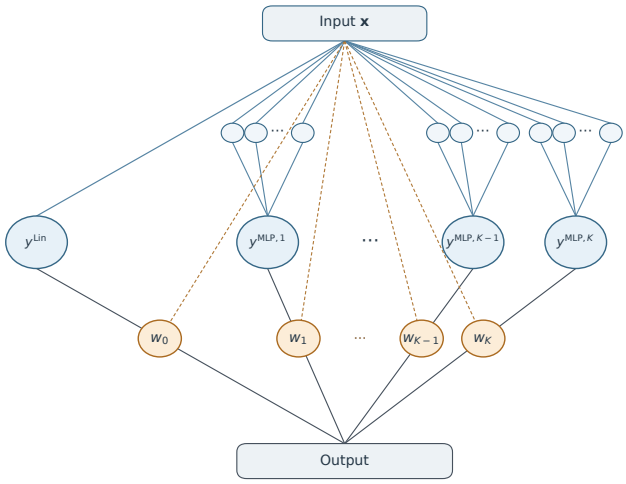

Figure 1: Schematic of EliMoE architecture with heterogeneous expert branches.

is weighted by the parameter γ, which controls the contribution of the global model prediction. The auxiliary loss $\mathcal{L}_{\text{aux}}$ captures the individual performance of the experts. The first term in $\mathcal{L}_{\text{aux}}$ corresponds to the error of the linear expert. The next K terms correspond to masked losses of the MLP experts.

The proposed objective extends the standard MoE training objective by introducing an auxiliary expert-level loss. This modification is motivated by a specific optimization issue in conventional MoE training. In particular, the parameters θ_j of expert j are updated according to the gradient of the overall loss with respect to θ_j. Under the standard mixture objective, this gradient can be written as

$$\nabla_{\theta_j} L_{\text{base}} = w_j(\mathbf{x}) \frac{\partial L_{\text{base}}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}_j}{\partial \theta_j}.$$

This expression shows that the learning signal received by expert j is directly scaled by its gating weight. Consequently, when $w_j(\mathbf{x})$ is small, the corresponding expert receives only a weak gradient and may therefore learn slowly or remain undertrained.

Under the proposed objective, the gradient with respect to the parameters θ_j of expert k is given by

$$\nabla_{\theta_j} L = \gamma \nabla_{\theta_j} L_{\text{base}} + (1-\gamma) \nabla_{\theta_k} L_{\text{aux}}.$$

Unlike the gradient of the base loss, $\nabla_{\theta_j} L_{\text{aux}}$ is not scaled by the current gating weight. The auxiliary loss therefore provides each expert with a direct supervision signal from the target, including experts that are temporarily assigned low routing probabilities. Hence, including the auxiliary loss addresses the weak-gradient problem caused by small gating weights.

In addition to improving the optimization process, the auxiliary loss in equation 1 encourages experts to learn from different temporal segments of the data by applying an expert-specific masking strategy. In particular, the red terms in $\mathcal{L}_{\text{aux}}$ determine the subset of observations used to train each expert, such that different experts are exposed to different portions of the training window, thereby giving each expert a distinct temporal learning task. Experts trained on shorter segments focus more strongly on recent observations and can adapt to short-term changes, whereas experts trained on longer segments use a broader history and can capture more persistent patterns.

It is worth highlighting that the proposed objective function is independent of the architecture of the individual experts. In this study, we employ linear and MLP experts, as our primary objective is to develop a model that improves forecasting accuracy without introducing excessive computa-

tional overhead. Nevertheless, the proposed MoE framework is sufficiently flexible to accommodate other types of experts. Additionally, under a rolling-window forecasting scheme, a computationally efficient model architecture alone is not sufficient to ensure fast training, since the model must be repeatedly updated after each forecasting step. To address this challenge, we combine the proposed MoE framework with the partial online learning strategy described in Section 4.2.

4 EXPERIMENTS

4.1 FIXED-SCHEME FORECASTING STUDY

We conduct a fixed-scheme forecasting study using five datasets from diverse fields with varying frequencies. These datasets are extracted from Monash Time Series Forecasting Repository Godahewa et al. (2021). The first dataset is the electricity demand of 321 individuals originally sourced from the UCI repository UCI Machine Learning Repository (2020). We use the aggregated weekly series for our analysis. In addition to this energy dataset, we also include the hourly M4 competition dataset Makridakis et al. (2018; 2020), which comprises 100,000 time series. Furthermore, we use the monthly tourism dataset, consisting of 1,311 tourism-related time series originating from a Kaggle competition Athanasopoulos et al. (2011); Ellis (2018), as well as a monthly car parts dataset comprising 2,674 time series representing sales data. The latter dataset is originally provided in the expsmooth R package Hyndman (2015) and contains missing values. In this study, we use the version where missing values have been imputed. The final dataset is the Saugeen dataset, which consists of a single long time series with 23,741 daily observations of the mean flow of the Saugeen River. This dataset is sourced from the deseasonalize R package McLeod & Gweon (2013).

To generate forecasts for each of the datasets described above, we employ EliMoE comprising four experts: one linear expert and three MLP experts. The model uses only lagged values of the target series as regressors. For each dataset, the number of lags and the forecast horizon are set according to the specifications used in Godahewa et al. (2021). The forecasting accuracy and computational efficiency of EliMoE are then compared with those of several benchmark models, including the feed-forward neural network (FFNN, Goodfellow et al., 2016), DeepAR Salinas et al. (2020), N-BEATS Oreshkin et al. (2020), WaveNet Borovykh et al. (2017), the Transformer Vaswani et al. (2017), PatchTST Nie et al. (2022). In addition, we consider two MoE-based benchmarks: TimeMoE, a zero-shot time-series foundation model Shi et al. (2025), and Mixture-of-Linear-Experts (MoLE) with an RMLP backbone Li et al. (2023); Ni et al. (2024).

The forecast comparison between the proposed model and the benchmark models is based on two popular error metrics: Mean Absolute Error (MAE, mae, 2010) and Root Mean Squared Error (RMSE). In addition, Mean Absolute Scaled Error (MASE, Hyndman & Koehler, 2006) is considered, as it provides scale-independent evaluation and accounts for seasonality when a seasonal naïve benchmark is used. Since most of the dataset contains multiple time series, we report mean values of each evaluation metric to summarize overall performance.

To ensure a rigorous experimental setup, we reran EliMoE and all baseline models 10 times using different random seeds and report the mean and standard deviation of the respective performance metrics and computational runtime. Table 1 reports the mean MASE across the 10 runs, together with the average computational time required for model fitting and forecast generation.

The results in Table 1 show that the EliMoE achieves strong forecasting performance across the considered datasets. It attains the lowest MASE on the Electricity Weekly and Saugeen River flow datasets. On the Carparts dataset, its performance is comparable to that of the best-performing models, including FFNN, DeepAR, and Transformer. Although EliMoE does not achieve the lowest MASE on the Tourism Monthly and M4 Hourly datasets, it attains the second-best performance in both cases, demonstrating its competitiveness across a diverse set of forecasting tasks.

The superiority of the proposed model remains evident when alternative error metrics are considered, as shown in Tables 4 and 5 in Appendix B. Although EliMoE does not achieve the best performance on every dataset under all evaluation metrics, it consistently exhibits competitive performance, placing it among the top-performing models overall.

In addition to its strong forecasting performance, EliMoE is the fastest model across all datasets except M4 Hourly, where it achieves the second-lowest computational time. The computational

Table 1: Comparison of forecasting accuracy and computational cost across datasets and models. Each model is represented by two rows: the first reports the mean MASE, and the second reports the mean runtime in seconds, with the corresponding standard deviations shown in parentheses. For each dataset, the lowest mean MASE and the fastest mean runtime are highlighted in bold. The smallest standard deviations for MASE and runtime are underlined across all models.

Model	Electricity weekly	M4 hourly	Saugeen day	Tourism monthly	Car parts
EliMoE	**0.76** (0.01)	1.47 (0.05)	**1.40** (0.01)	1.49 (0.01)	**0.75** (0.00)
	3.08 (0.51)	35.52 (0.79)	**2.14** (0.12)	**8.98** (0.65)	**3.37** (0.30)
MoLE	0.80 (0.04)	3.53 (0.66)	1.55 (0.05)	2.05 (0.07)	0.94 (0.01)
	20.09 (3.88)	299.93 (62.30)	18.67 (0.36)	70.64 (0.67)	28.76 (0.57)
PatchTST	0.90 (0.04)	3.11 (0.16)	1.58 (0.02)	1.77 (0.03)	0.93 (0.01)
	274.83 (5.59)	9560.95 (28.53)	96.73 (0.97)	409.28 (17.26)	159.94 (4.09)
TimeMoE*	1.09(–)	**0.88**(–)	2.13(–)	2.88(–)	1.07(–)
	4.85(–)	22.54(–)	0.98(–)	4.28(–)	27.54 (–)
DeepAR	1.17 (0.12)	2.57 (0.76)	1.92 (0.20)	**1.44** (0.03)	**0.75** (0.00)
	334.51 (19.37)	1302.47 (21.64)	255.93 (12.36)	179.53 (16.53)	169.70 (2.15)
FFNN	0.80 (0.05)	2.67 (0.61)	1.42 (0.04)	1.61 (0.03)	**0.75** (0.00)
	15.71 (1.03)	**23.61** (1.24)	40.42 (4.60)	14.73 (1.45)	16.64 (0.29)
N-BEATS	0.78 (0.04)	1.91 (0.34)	1.87 (0.08)	1.57 (0.03)	2.84 (0.00)
	1245.82 (85.93)	1443.99 (5.14)	1203.38 (34.46)	1302.66 (12.49)	1491.96 (147.88)
Transformer	1.33 (0.15)	4.87 (1.82)	1.87 (0.06)	1.79 (0.48)	**0.75** (0.00)
	277.13 (0.47)	1620.51 (15.33)	212.45 (1.80)	146.91 (7.50)	150.79 (0.63)
WaveNet	1.11 (0.09)	1.80 (0.20)	1.52 (0.06)	1.49 (0.03)	0.76 (0.02)
	613.13 (51.49)	2939.62 (283.52)	946.30 (55.27)	696.03 (51.21)	334.39 (29.45)

*TimeMoE runtime includes only data loading and forecast generation. A value of (-) indicates that the standard deviation is zero due to the deterministic nature of TimeMoE.

advantage of EliMoE over the other benchmark models is often substantial. For instance, on the Car Parts dataset, EliMoE requires less than four seconds for model training and forecast generation, while the second-fastest model requires approximately four times as much computational time. Note that TimeMoE is excluded when identifying the fastest model. Since TimeMoE is a pretrained model, the computational time reported in Table 1 includes only model loading and forecast generation.

Another noteworthy result in Table 1 is the low variability of EliMoE across different random seeds. Across the considered datasets, EliMoE generally achieves the smallest standard deviation for both MASE and computational time, suggesting that its forecasting accuracy and runtime are relatively stable across repeated runs.

4.2 Online learning

The forecasting study in the previous section employed the fixed scheme, in which the model is trained once. The same weights are used to generate forecasts over the entire horizon. This approach is computationally inexpensive but often comes at the cost of reduced accuracy, particularly for longer forecast horizons. In time series analysis, there are other forecasting approaches that yield more accurate forecasts, such as the rolling window scheme. In this approach, the model is repeatedly retrained using a moving training window. However, this repeated retraining is computationally very expensive, particularly since each subsequent training window differs from the previous one by only a single additional observation.

To minimize the computational burden introduced by the rolling window scheme, we combine the proposed MoE with partial online learning introduced by El Mahtout & Ziel (2026). This approach reduces training time through two key mechanisms. First, it employs warm starting, where model weights from the previous training window are used to initialize the current window, thereby accelerating convergence. Second, it uses two distinct sets of hyperparameters: one optimized for the initial training window and another shared across all subsequent windows.

We conduct a rolling-window forecasting study on the Saugeen River Flow dataset, combining EliMoE with the partial online learning strategy. The forecasting accuracy and execution time of the resulting model are then compared with those of the FFNN and MoLE benchmarks. EliMoE and both benchmark models are hyperparameter-tuned using the Tree-structured Parzen Estimator (TPE) optimization method Bergstra et al. (2011), implemented through the Python package Optuna Akiba et al. (2019). Further details on the selected hyperparameters and the validation dataset are provided in Appendix C.

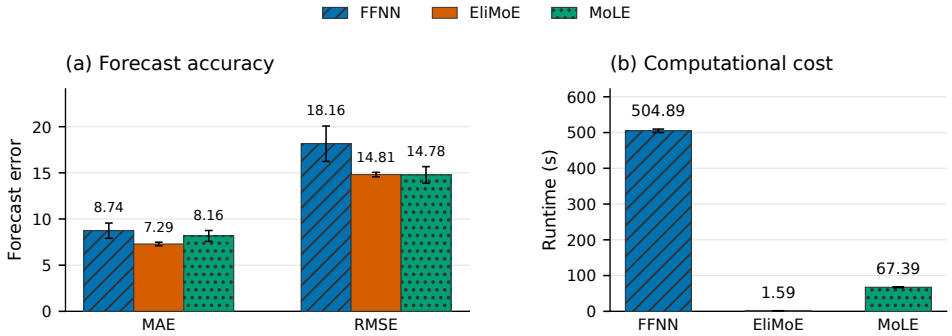

Figure 2: Forecast accuracy and computational cost of FFNN, EliMoE, and MoLE on the Saugeen River Flow dataset under rolling window scheme. Bars report the mean MAE, RMSE, and runtime across ten random seeds, with error bars indicating $\pm$ one standard deviation.

During hyperparameter tuning, the best-performing set of hyperparameters is selected for each model and subsequently used to generate forecasts on the test data across ten different random seeds. Figure 2 illustrates the average MAE, RMSE, and computational time across the ten seeds for the considered models. As expected, all models demonstrate substantially improved performance compared with the fixed-scheme framework. For example, the MAE of the FFNN decreases from 21.39 under the fixed scheme (see Table 4 in the Appendix) to 8.74 under the rolling window strategy. Similar improvements are also observed for EliMoE and MoLE.

Within the rolling-window framework, EliMoE clearly outperforms the FFNN, reducing MAE by 17% and RMSE by 18%. Compared with MoLE, EliMoE achieves an 11% reduction in MAE, while RMSE increases marginally, by less than 1%. EliMoE also demonstrates substantially higher computational efficiency, requiring approximately two seconds to complete the full forecasting study, compared with approximately eight minutes for the FFNN and one minute for MoLE. Additionally, the error bars indicate that EliMoE exhibits lower variability across the ten random seeds than the other models under the rolling-window scheme, which is consistent with the results observed under the fixed forecasting scheme.

4.3 Ablation Study

To assess the effectiveness of the proposed framework, we conduct four experiments designed to isolate the contributions of its main components. Specifically, we investigate the effect of the weighting coefficient γ, the impact of the masking strategy, the effect of the number of experts on forecasting accuracy and computational time, and the generalizability of the proposed framework. All experiments are conducted using the Saugeen River Flow dataset under the rolling-window forecasting scheme, except for the analysis of the impact of the number of experts, which is conducted under the fixed forecasting scheme.

The parameter γ in the objective function defined in Equation 1 is a crucial component of the proposed methodology. It takes values in the interval [0,1]. When $\gamma = 0$, only the expert-specific losses contribute to the training process, while the aggregated forecasting loss is ignored. In contrast, when $\gamma = 1$, the expert-specific losses no longer participate in training, and the model reduces to the standard formulation based solely on the base loss. Note that throughout this study, we set $\gamma = 0.5$, thereby assigning equal weight to the base loss and the expert-specific losses.

The first two rows of Table 2 illustrate the impact of different values of γ on the forecasting accuracy of EliMoE. For $\gamma \in \{0.25, 0.5, 0.75\}$, MAE decreases by approximately 14-24% relative to $\gamma = 1$,

which corresponds to the standard MoE setting. Similarly, RMSE improves by approximately 2-15%. These results indicate that incorporating expert-specific losses together with the masking strategy can substantially improve forecasting accuracy compared with relying solely on the base loss.

Table 2: Effect of γ and masking on EliMoE online forecasting accuracy on the Saugeen day dataset. Values are means over 10 seeds. Bold indicates the lowest error in each row

Setting	**Metric**	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
With masking	MAE	8.90	7.88	**7.29**	8.25	9.56
	RMSE	17.43	15.47	**14.81**	16.99	17.37
Without masking	MAE	**7.97**	8.31	8.65	8.50	9.56
	RMSE	**16.08**	16.85	17.03	17.10	17.37

Expert masking is an important component of the proposed methodology, as it encourages different experts to learn from different temporal segments of the training data. In the EliMoE specification used in this study, the model consists of one linear expert and three MLP experts. When computing the auxiliary expert-specific losses, the linear expert and MLP^1 use the full training window, whereas MLP^2 uses the most recent two-thirds of the observations and MLP^3 uses only the most recent one-third. This nested partitioning ensures that all experts have access to the most recent observations. Note that, when computing the base loss, all experts have access to the full training window.

To isolate the effect of the masking mechanism, we retain the auxiliary expert-specific losses in the objective function but remove the masking strategy. In this setting, all experts use the full training window when computing their auxiliary losses. The last rows of Table 2 report the results obtained without masking for different values of γ. Comparing these results with their corresponding masked configurations shows a clear deterioration in forecasting accuracy when masking is omitted for $\gamma \in \{0.25, 0.5, 0.75\}$. This finding provides evidence that the masking mechanism contributes to the forecasting performance of EliMoE. An exception is observed for $\gamma = 0$, where forecasting performance improves when masking is removed. In this setting, the base loss is completely omitted, and training is driven exclusively by the auxiliary expert-specific losses.

The number of experts is another crucial component of the proposed method, particularly when computational efficiency is taken into consideration. To investigate the trade-off between forecasting accuracy and computational cost, we evaluate EliMoE with different numbers of MLP experts and report the corresponding MAE and runtime. Figure 3 summarizes the results.

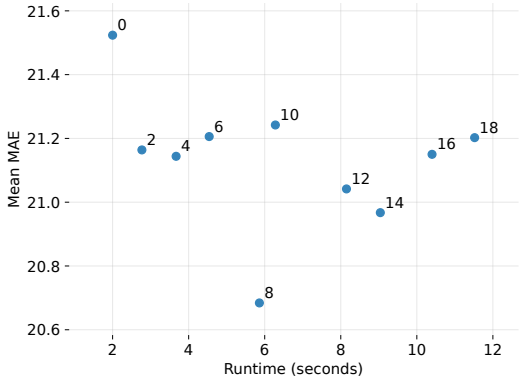

Figure 3: Forecast accuracy–runtime trade-off of EliMoE for different numbers of experts. Each point represents one model configuration, with the annotation indicating the corresponding number of MLPs.

When the number of MLP experts is zero, EliMoE consists solely of the linear expert, resulting in a relatively high MAE but a very short runtime. As the number of MLP experts increases, MAE

generally decreases, while computational time increases. The lowest MAE is achieved with eight MLP experts and one linear expert. Beyond this point, increasing the number of experts does not yield further improvements in forecasting accuracy, while computational cost continues to increase.

Finally, to evaluate whether the proposed methodology generalizes to other MoE architectures, we incorporate the expert-specific loss into the objective function of MoLE. Table 3 reports the resulting forecasting performance for different values of γ.

Table 3: Forecasting performance of MoLE augmented with the expert-specific loss for different values of γ on the Saugeen River Flow dataset. Results are averaged across ten random seeds, and bold values indicate the best performance for each metric.

Metric	$\gamma = 0$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.75$	$\gamma = 1$
MAE	7.33	7.74	7.45	**7.19**	8.16
RMSE	**13.85**	14.42	14.15	14.13	14.78

The case $\gamma = 1$ corresponds to the standard MoLE formulation used earlier, in which training relies solely on the base loss. Compared with this setting, incorporating the expert-specific loss consistently improves forecasting accuracy, including the case $\gamma = 0$, where training is driven entirely by the auxiliary loss. The maximum improvement reaches approximately 12% for MAE and 6% for RMSE. These results provide evidence that the proposed expert-specific loss is not limited to EliMoE and can also improve the forecasting performance of other MoE architectures.

5 Conclusion

In this work, we introduced a novel MoE framework for time series forecasting aimed at improving forecasting accuracy while reducing computational cost. The proposed EliMoE incorporates expert-specific losses into the objective function in addition to the aggregated forecasting loss, enabling expert-level prediction errors to directly influence the training process. Additionally, masking is applied to the MLP-specific loss to encourage the network to specialize in a specific subset of the data.

EliMoE demonstrates strong forecasting performance across diverse time series domains, outperforming a broad set of established and state-of-the-art neural forecasting models, including DeepAR, N-BEATS, WaveNet, PatchTST, TimeMoE, and MoLE. The model also achieves substantially higher computational efficiency than the baseline methods. In addition, the proposed methodology is flexible and can be incorporated into other MoE architectures. Empirical results with MoLE show that integrating the proposed expert-specific loss improves forecasting accuracy, providing further evidence of the broader applicability of the methodology.

Finally, it is important to acknowledge several limitations of the present study. First, only five values of the γ parameter in the objective function were considered, leaving the sensitivity of the proposed objective to a finer range of weighting choices unexplored. In addition, the nested masking strategy ensures that all experts retain access to the most recent observations and has been shown empirically to improve forecasting accuracy. However, there is no guarantee that this masking design is optimal. Future research should therefore investigate the effect of γ more comprehensively and explore alternative masking strategies, including learnable temporal partitions.

DECLARATION ON AI-ASSISTED TOOLS

During the preparation of this work, the authors used generative AI tools to improve the readability and clarity of the manuscript, assist with code editing and debugging, and support the literature review. All AI-assisted content and code were reviewed, verified, and edited by the authors as needed. The authors take full responsibility for the final content of the manuscript.

REFERENCES

Mean absolute error. In Claude Sammut and Geoffrey I. Webb (eds.), *Encyclopedia of Machine Learning*, pp. 652–652. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

George Athanasopoulos, Rob J Hyndman, Haiyan Song, and Doris C Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

G.E.P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. HoldenDay Inc., 1990.

Btissame El Mahtout and Florian Ziel. Electricity price forecasting: Bridging linear models, neural networks and online learning. *arXiv preprint arXiv:2601.02856*, 2026.

Peter Ellis. Tcomp: Data from the 2010 tourism forecasting competition. `https://cran.r-project.org/web/packages/Tcomp`, 2018.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

Rob J. Hyndman. expsmooth: Data sets from forecasting with exponential smoothing. `https://cran.r-project.org/web/packages/expsmooth`, 2015. R package.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 3rd edition, 2021. URL `https://otexts.com/fpp3/`.

Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations (ICLR)*, 2021.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6265–6274. PMLR, 2021.

Zhe Li, Shiyi Qi, Yiduo Li, and Zenglin Xu. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. *arXiv preprint arXiv:2410.10469*, 2024.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of forecasting*, 34(4):802–808, 2018.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020.

AI McLeod and Hyukjun Gweon. Optimal deseasonalization for monthly and daily geophysical time series. *Journal of Environmental statistics*, 4(11):1–11, 2013.

Ronghao Ni, Zinan Lin, Shuaiqi Wang, and Giulia Fanti. Mixture-of-linear-experts for long-term time series forecasting. In *International conference on artificial intelligence and statistics*, pp. 4672–4680. PMLR, 2024.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations (ICLR)*, 2025. Spotlight presentation; preprint arXiv:2409.16040.

UCI Machine Learning Repository. Electricityloaddiagrams2011–2014 data set. `https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014`, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Kyrylo Yemets, Ivan Izonin, Mariia Zimokha, and Michal Gregus. Load-balancing strategies for forecasting with mixture-of-experts architecture. *Procedia Computer Science*, 2025. `https://www.sciencedirect.com/science/article/pii/S1877050925035380`.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Lingzheng Zhang, Lifeng Shen, Yimin Zheng, Shiyuan Piao, Ziyue Li, and Fugee Tsung. Lemole: Llm-enhanced mixture of linear experts for time series forecasting. *arXiv preprint arXiv:2412.00053*, 2024.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Quoc V. Le, James Laudon, and Zhifeng Chen. Mixture-of-experts with expert choice routing. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. Taming sparsely activated transformer with stochastic experts. In *International Conference on Learning Representations (ICLR)*, 2022.

A Appendix

B Tables

Table 4: Comparison of forecasting accuracy across datasets and models in terms of MAE, reported as the mean with the standard deviation in parentheses. For each dataset, the lowest mean and lowest standard deviation are highlighted independently in bold.

Model	Electricity weekly	M4 hourly	Saugeen day	Tourism monthly	Car parts
EliMoE	30501.05 (**1571.66**)	**273.60** (**7.48**)	**21.18** (**0.19**)	**1809.03** (**20.63**)	**0.39** (**0.00**)
MoLE	**27760.39** (5553.18)	330.17 (63.95)	23.32 (0.73)	3263.75 (305.33)	0.50 (0.01)
PatchTST	35848.88 (3955.40)	342.23 (13.78)	23.85 (0.27)	2236.85 (62.37)	0.49 (0.01)
TimeMoE	55794.93 (–)	301.28 (–)	32.09 (–)	4431.75 (–)	0.58 (–)
DeepAR	54329.90 (10730.86)	467.57 (127.61)	28.92 (3.01)	2032.93 (182.12)	**0.39** (**0.00**)
FFNN	35222.10 (3571.31)	378.59 (45.34)	21.39 (0.62)	2086.13 (133.36)	**0.39** (**0.00**)
N-BEATS	29719.87 (4627.87)	312.49 (45.54)	28.12 (1.19)	1974.64 (63.87)	0.98 (**0.00**)
Transformer	66064.88 (11599.76)	605.79 (320.80)	28.26 (0.89)	2743.05 (1016.36)	**0.39** (**0.00**)
WaveNet	63412.47 (6617.03)	382.35 (47.49)	22.90 (0.89)	2302.68 (342.33)	0.40 (0.01)

Table 5: Comparison of forecasting accuracy across datasets and models in terms of RMSE, reported as the mean with the standard deviation in parentheses. For each dataset, the lowest mean and lowest standard deviation are highlighted independently in bold.

Model	Electricity weekly	M4 hourly	Saugeen day	Tourism monthly	Car parts
EliMoE	33309.88 (**1567.32**)	**330.68** (**10.46**)	40.17 (0.26)	**2256.25** (**20.22**)	0.74 (**0.00**)
MoLE	197737.78 (69124.33)	1452.09 (418.75)	**39.08** (1.35)	25032.51 (4894.43)	1.09 (**0.00**)
PatchTST	39449.42 (3759.32)	412.52 (14.82)	41.59 (**0.19**)	2826.11 (80.71)	**0.70** (**0.00**)
TimeMoE	58824.94 (–)	391.25 (–)	39.92 (–)	5658.86 (–)	0.77 (–)
DeepAR	57261.98 (10600.88)	566.57 (137.46)	50.02 (2.36)	2586.55 (216.79)	0.74 (**0.00**)
FFNN	38214.13 (3541.06)	454.88 (48.60)	41.16 (1.25)	2586.74 (139.89)	0.74 (**0.00**)
N-BEATS	**32986.41** (4201.02)	382.52 (52.59)	49.31 (0.74)	2551.90 (77.67)	1.12 (**0.00**)
Transformer	68567.70 (11671.59)	680.92 (321.31)	49.19 (0.81)	3301.36 (1098.47)	0.74 (**0.00**)
WaveNet	66187.50 (6605.94)	470.08 (54.43)	43.82 (1.39)	2878.62 (400.63)	0.75 (0.01)

C FORECAST STUDY

We conduct a fixed-window forecasting study using lagged values of the target variable as predictors. The number of lags and the forecast horizon depend on the dataset. For instance, for the Saugeen dataset, we use 9 lagged observations and a forecast horizon of 30. The lag structure and forecast horizons for all datasets follow the configurations proposed by Godahewa et al. (2021).

EliMoE was implemented using PyTorch. We used the same hyperparameters as the FFNN model, such as the learning rate. However, different values were selected for the batch size and number of epochs, since the FFNN was implemented using GluonTS and required a different training configuration. The selected hyperparameters are summarized in the following table.

Table 6: Hyperparameter configuration for the MoE and FFNN models

Hyperparameter	**MoE**	**FFNN (GluonTS)**
Learning rate	0.001	0.001
Weight decay	1e-8	-
Batch size	256	32
Number of epochs	100	100
Number of batches per epoch	-	50
Optimizer	Adam	Adam
Framework	PyTorch	GluonTS

In the rolling-window forecasting study, the Saugeen dataset is used. The last 30 observations are kept untouched as the test set, while the 30 observations preceding the test set are allocated to a validation set for hyperparameter tuning.

For all models, the hyperparameter settings from the fixed-scheme study are retained, with only three hyperparameters being tuned: the number of neurons, the learning rate, and the window size. The following table summarizes the search space considered for these parameters.

Table 7: Tuned hyperparameters for benchmarks and EliMoE in the rolling-window forecasting study.

Hyperparameter	**FFNN**	**EliMoE (Initial Window)**	**EliMoE (Updated Window)**	**MoLE**
Number of neurons	[1, 50]	[1, 50]	[1, 50]	-
Learning rate	[1e-5, 1e-2]	[1e-5, 1e-2]	[1e-4, 1e-2]	[1e-5, 1e-2]
Window size	[730, 3650]	[730, 3650]	[1, 1095]	[730, 3650]
Number of hidden units	-	-	-	[8, 512]

Note that all calculations for the proposed and benchmark models were conducted on a MacBook Pro (2020) equipped with an Apple M1 processor (8 CPU cores) and 16 GB of unified memory.