FUZZY-MOE: INTERPRETABLE REGIME-CONDITIONED EXPERT ROUTING FOR NON-STATIONARY MULTIVARIATE TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

In non-stationary multivariate time series, different variables and samples often exhibit heterogeneous latent dynamic states, while existing deep forecasting models usually compress them into a unified end-to-end mapping, leading to suboptimal modeling of time-varying dynamics and limited interpretability regarding which forecasting mechanism is activated under different latent states. To overcome these limitations, we reformulate time series forecasting as a unified framework of latent temporal state identification and interpretable expert routing, and propose Fuzzy-MoE, a fuzzy logic-based dynamic Mixture-of-Experts model. Fuzzy-MoE consists of multiple parallel expert mapping networks and a dual-view fuzzy router. By jointly exploiting local convolutional dynamics and global segmented statistics, the router infers latent temporal states and computes expert activation strengths through learnable Gaussian membership functions, enabling explicit IF-THEN rule-based expert selection. This fine-grained routing strategy allows different variables within the same sequence to activate different experts, effectively capturing heterogeneous temporal dynamics while improving model interpretability. Experimental results on multiple public time series benchmark datasets show that Fuzzy-MoE significantly outperforms mainstream forecasting methods in forecasting accuracy. Moreover, fuzzy memberships and rule activations provide interpretable routing diagnostics, demonstrating the effectiveness of the proposed framework in both forecasting performance and mechanism transparency. Unlike traditional MoE models that use black-box routing, Fuzzy-MoE's routing is based on clear, interpretable fuzzy rules. This makes the expert selection transparent and traceable.

1 INTRODUCTION

Non-stationary multivariate time series forecasting is fundamental to numerous real-world applicationsJin et al. (2024); Gruver et al. (2023); Liang et al. (2024), yet its inherent heterogeneity remains a primary obstacle. In such data, the underlying temporal dynamics often vary not only across different time segments (sample-level) but also across different variables within the same observation window (channel-level). On non-stationary multivariate series, a single global mapping trained to minimize the average loss tends to smooth over local, sample- and channel-specific dynamics, leading to suboptimal fits on the segments where those dynamics dominate Wu et al. (2021); Zeng et al. (2023); Nie et al. (2023); Liu et al. (2024); Weng et al. (2025); Murad et al. (2025), as the model fails to explicitly identify which latent state governs the current input Sun et al. (2025).

To cope with this heterogeneity, recent works have turned to Mixture of Experts (MoE) architectures, which employ multiple expert networks to cover different pattern subspaces. However, the routing mechanisms in these models, which determine how inputs are assigned to experts, remaining a critical bottleneck Yang et al. (2025); Wang et al. (2026). The vast majority of existing predictors rely on black-box linear projections followed by Softmax gating, as shown in Figure 1(a). Such designs not only lack interpretability, as the resulting weights offer no insight into which temporal features drive the selection of a particular expert, but they are also prone to gate-collapsing, where

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.

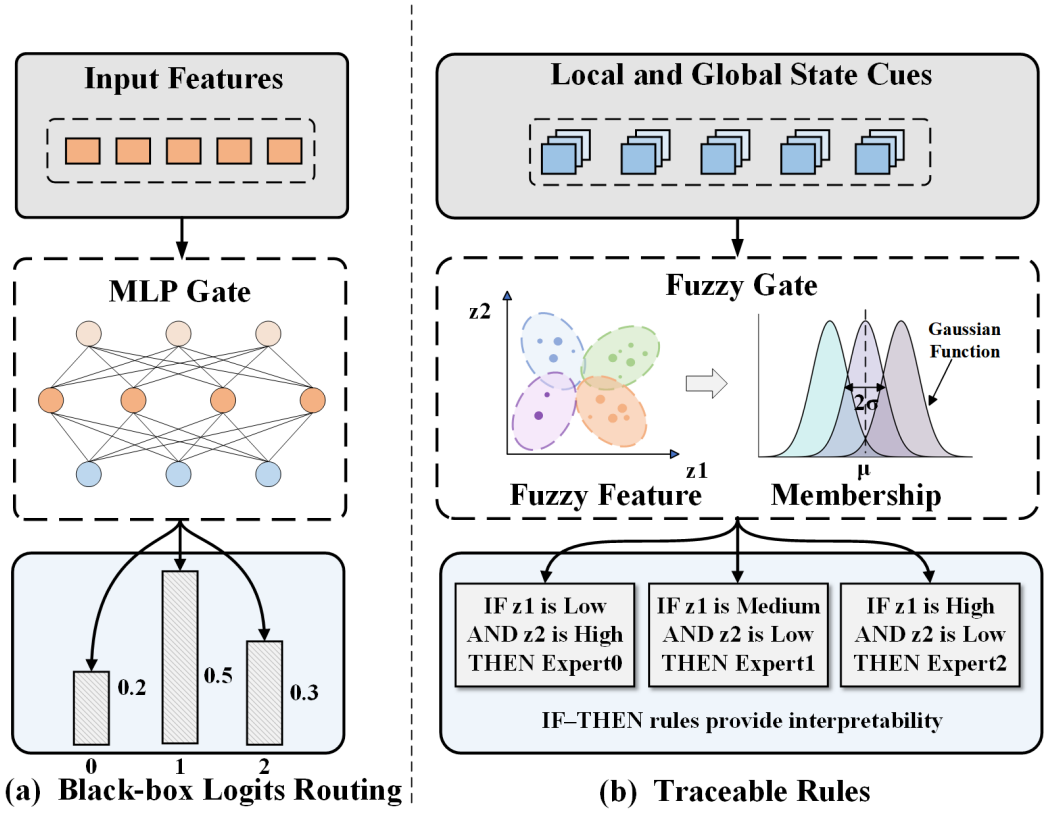


Figure 1: Comparison of classic black-box MLP gate and our interpretable fuzzy gate. (a) Traditional MLP gate generates unexplainable routing logits without traceable decision rules; (b) Our fuzzy gate fuses local and global temporal cues, computes expert weights via Gaussian fuzzy membership, and yields readable IF-THEN routing rules for fine-grained variable-wise expert assignment.

a few experts dominate all inputs while others degenerate. Consequently, despite strong fitting capabilities, these models provide no transparent rationale for their decisions, severely limiting trust and deployability in high-stakes domains. Standard MoE has a problem. It outputs routing weights but cannot explain why. It cannot answer: 'Why was this expert chosen?' This makes the model hard to trust.

We address these challenges by fundamentally reformulating the forecasting task. Instead of learning a single direct mapping, we propose a paradigm shift toward latent temporal state identification followed by interpretable expert routing. Our core philosophy is that the model should first diagnose the underlying dynamic regime of each input, and then explicitly select the expert network best suited to that regime. This decomposition transforms the opaque end-to-end forecasting into a two-stage, inspectable process: state reasoning and conditional forecasting. Our method is different from standard MoE. We do not hide routing weights inside. Instead, we create clear IF-THEN rules. You can see which expert is chosen and why.

In this work, we proposed Fuzzy-MoE, a fuzzy logic-based dynamic Mixture-of-Experts model, as shown in Figure 1(b). It extracts complementary state cues via a dual-view router that captures local convolutional dynamics and global segmented statistics. These cues are projected into a low-dimensional fuzzy space, where routing weights are generated through learnable Gaussian membership functions and explicit IF-THEN rule firings. Crucially, the router operates at the sample-channel level, allowing different variables within the same sequence to activate different experts, a fine-grained adaptability that mirrors real-world heterogeneity. The product T-norm aggregation and temperature-scaled Softmax ensure numerical stability while preserving the physical meaning of each rule's activation, yielding a fully traceable decision path. Each rule is simple: IF fuzzy vari-

able is in range $[\mu - \sigma, \mu + \sigma]$, THEN activate expert$_k$. You can understand the rule without seeing the weights. In summary, our contributions are as follows:

- We redefine non-stationary multivariate forecasting from the traditional sequence-to-sequence mapping to an interpretable state-conditioned expert routing problem. This formulation extends the contribution beyond a single forecasting task, pointing to a more general adaptive reasoning paradigm under heterogeneous data dynamics.
- We propose a transparent routing architecture that extracts local and global state cues, and generates expert weights through learnable Gaussian membership functions and explicit IF-THEN rules, making latent state identification an inspectable and interpretable intermediate process.
- Our model assigns experts at the sample-channel level, thereby enabling different variables within the same sequence to invoke distinct forecasting mechanisms. Furthermore, the framework produces multi-level diagnostic signals, including fuzzy memberships, expert weights, and rule activations, which collectively reveal the model's confidence, uncertainty, and the specific rationale underlying each expert selection.

2 Related Works

2.1 Time Series Forecasting

Time series forecasting is a core problem in the field of data mining and has received widespread attention for a long time Qiu et al. (2026); Wu et al. (2026). Traditional methods, represented by autoregressive moving average (ARIMA) perform well when dealing with stationary linear data but lack the ability to fit nonlinear complex patterns. Machine learning methods like support vector regression (SVR) improve forecasting power through nonlinear kernels or ensemble strategies, they still rely on manually designed features and struggle to automatically uncover deep temporal representations. In recent years, deep learning models have made significant progress in time series forecasting. Transformer-based models leverage self-attention mechanisms to capture global dependencies and have become one of the mainstream paradigms. Representative works include: PatchTST Nie et al. (2023) and iTransformer Liu et al. (2024), which use an inverted architecture embedding independent variable sequences into the attention mechanism to better model multivariate correlations. TimesNet Wu et al. (2023) transforms 1D time series into 2D tensors via Fourier transform, capturing both intra- and inter-period variations. Autoformer Wu et al. (2021) replaces standard self-attention with an autocorrelation mechanism to explicitly model sequence periodicity. FEDformer Zhou et al. (2022) introduces frequency-domain enhancement modules to improve long-term forecasting performance. However, all these models rely on a single mapping network to convert from past to future Huang et al. (2025); Ma et al. (2025); Gao et al. (2024), making it difficult to adapt to heterogeneous dynamic states across different variables and segments in non-stationary time series.

2.2 Explainability in Time Series Forecasting

The black-box nature of deep models has long limited their use in high-risk areas like finance and healthcare Ma et al. (2025); Jiang et al. (2025); Hertel et al. (2026). Existing research on interpretability mainly follows two paths Mehouachi & Jabari (2025). The first is post-hoc explanation, such as analyzing a trained model's behavior using methods like attention weight visualization. However, post-hoc explanations often differ from the model's actual decisions and can even be misleading. The second path is inherently interpretable models Wang et al. (2025), which make the reasoning process transparent through self-explanatory structures. Temporal Fusion Transformer (TFT) Lim et al. (2021) provides decision support via static variable encoding and interpretable multi-head attention. While these methods do offer some level of explanation, their interpretations usually stop at the attribution level of which inputs are important, without addressing the deeper question of under what conditions does the model use which forecasting mechanism.

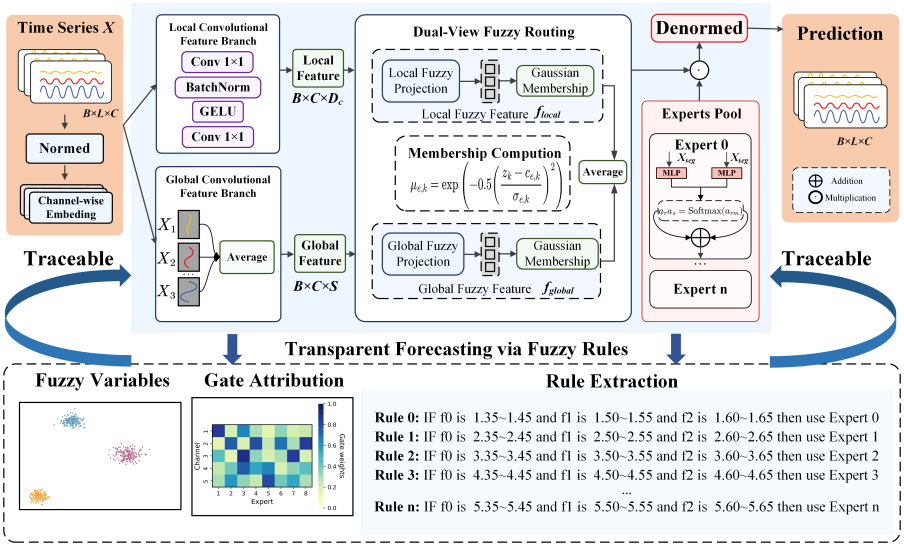


Figure 2: Overall pipeline of the proposed Fuzzy-MoE framework. It integrates input segmentation, dual-view state extraction, fuzzy routing, residual experts pool and output recovery. The dual-view fuzzy router outputs sample-channel-level expert weights, with auxiliary visualization modules to provide fuzzy embedding, gate attribution and explicit fuzzy rules for full routing interpretability.

2.3 Mixture of Experts

Mixture of Experts (MoE) Jiang et al. (2024) expands model capacity through the collaborative decisions of multiple expert networks, allowing different experts to handle different subsets of data. In recent years, it has shown remarkable results in large-scale pre-trained models. For example, Google's Switch Transformer Fedus et al. (2022) and GLaM Du et al. (2022) use MoE to scale model parameters to the trillion level while keeping computational costs nearly the same. In the field of time series forecasting, the MoE architecture has been explored preliminarily Zhang et al. (2025). Some works design different experts as recurrent or convolutional networks with varying receptive fields to handle multi-scale temporal patterns; others combine MoE with transformers, using sparse gating to select different attention heads or feedforward networks. However, the gating mechanism in existing MoE predictors is essentially a black-box linear projection followed by Softmax normalization Shi et al. (2025); Liu et al. (2025). They fail to generate sample-channel-level fuzzy conditional rules for fine-grained variable-wise expert allocation. Pattern-specific and distribution-shift-aware expert models mitigate temporal heterogeneity by grouping samples into distinct dynamic clusters and assigning dedicated experts per cluster, but their interpretability is limited to simple cluster-pattern matching without extracting complementary local convolutional and global segmented statistical state cues or mapping multi-scale temporal signals into traceable IF-THEN routing rules, which our Fuzzy-MoE specially designs for non-stationary multivariate forecasting tasks.

3 Proposed Method

In this section, we present the proposed Fuzzy-MoE framework in detail.

3.1 Overall: From Sequence Mapping to Regime-Conditioned Routing

Conventional deep forecasting models treat multivariate time series forecasting as an end-to-end sequence mapping problem: learning a unified function $f : \mathcal{X} \mapsto \mathcal{Y}$ that directly transforms the historical observation window into the future horizon. This "one-size-fits-all" paradigm implicitly assumes that a single set of parameters can adequately capture all temporal dynamic patterns. However, in non-stationary time series, different variables and different segments often reside in heterogeneous dynamic regimes, rendering this assumption invalid in practice.

In this paper, we reformulate the forecasting task as a **regime-conditioned expert routing** problem. The core idea is that the model should not directly learn a single mapping; instead, it should first identify the underlying temporal regime of the current input and then select the expert network that best suits that regime to complete the forecasting. Formally, given a historical sequence $\mathbf{X} \in \mathbb{R}^{L \times C}$ (where L is the look-back window length and C is the number of variables), the model extracts state clues from multiple views, and the fuzzy routing mechanism determines the expert selection:

$$\hat{\mathbf{Y}} = \sum_{k=1}^{K} \mathbf{w}_k \cdot \mathcal{E}_k(\mathbf{X}_{\text{seg}}), \tag{1}$$

where K is the total number of experts, $\mathbf{w}_k$ is the gating weight generated by the fuzzy router with explicit physical semantics, $\mathcal{E}_k$ denotes the k-th expert network, and $\mathbf{X}_{\text{seg}}$ is the segmented input. This formulation decomposes the forecasting process into two independently inspectable stages: **regime identification** (fuzzy routing) and **expert forecasting** (expert networks), laying the foundation for subsequent interpretability design.

3.2 Model Architecture Overview

The overall architecture of Fuzzy-MoE consists of five core modules: input normalization and segmentation, dual-view state feature extraction, dual-view fuzzy routing, expert mapping networks, and weighted fusion with output recovery. Figure 2 illustrates the complete model structure.

3.2.1 Input Normalization and Segmentation

To mitigate distribution shift, we first apply reversible instance normalization (RevIN) to the input sequence $\mathbf{X} \in \mathbb{R}^{B \times L \times C}$ (where B is the batch size):

$$\tilde{\mathbf{X}} = \frac{\mathbf{X} - \mu}{\sigma + \epsilon}, \quad \mu = \frac{1}{L}\sum_{t=1}^{L} \mathbf{X}, \quad \sigma = \sqrt{\frac{1}{L}\sum_{t=1}^{L} \left(\mathbf{X} - \mu\right)^2} \tag{2}$$

where ϵ is a small constant for numerical stability. The normalized sequence is then divided into $S = L/T_{\text{seg}}$ non-overlapping segments of length T_{seg}, yielding $\mathbf{X}_{\text{seg}} \in \mathbb{R}^{B \times C \times T_{\text{seg}} \times S}$. This segmentation reduces the sequence length while preserving local temporal structures, enabling efficient expert mapping at the segment level.

3.2.2 Dual-View State Feature Extraction

To comprehensively capture temporal state information, the model employs two complementary feature extraction pathways.

Local convolutional dynamics pathway: For each variable channel independently, a 1D convolutional network extracts short-term patterns and abrupt changes. This pathway consists of two convolutional layers: the first maps from 1 to 8 channels with kernel size k and stride s, followed by batch normalization and GELU activation; the second is a 1×1 convolution that maps back to 1 channel. The output is $\mathbf{H}_{\text{conv}} \in \mathbb{R}^{B \times C \times D_c}$, where D_c is the convolutional output dimension.

Global segment statistics pathway: We compute the mean value within each segment to capture long-term trends and periodic characteristics:

$$\mathbf{H}_{\text{global}} = \text{mean}_{T_{\text{seg}}}(\mathbf{X}_{\text{seg}}) \in \mathbb{R}^{B \times C \times S}, \tag{3}$$

where S is the number of segments. These two pathways focus on different temporal scales, providing complementary state clues for subsequent fuzzy routing.

3.3 Dual-View Fuzzy Routing Mechanism

The fuzzy router is the core component that enables interpretable expert selection in Fuzzy-MoE. It consists of two independent fuzzy gate modules: the convolutional fuzzy gate $\mathcal{G}_{\text{conv}}$ and the global

fuzzy gate $\mathcal{G}_{\text{global}}$. Each module's computation comprises three steps: fuzzy variable projection, Gaussian membership computation and rule activation, and gating weight generation. The final gating weights are obtained by fusing the outputs of the two pathways.

3.3.1 Fuzzy Variable Projection

Given an input feature $\mathbf{h} \in \mathbb{R}^D$ (where D is the feature dimension, with $D = D_c$ for the convolutional gate and $D = S$ for the global gate), we first project it into a low-dimensional interpretable fuzzy variable space via a multi-layer projection network:

$$\mathbf{z} = \text{Proj}(\mathbf{h}) \in \mathbb{R}^M, \quad M \ll D, \tag{4}$$

where the projection network Proj consists of two linear layers with LayerNorm and GELU activation in between, which can be seen in Figure 3. M is the number of fuzzy variables. This projection maps high-dimensional abstract features into a low-dimensional latent space. By constraining the fuzzy membership functions, the model forces the network to organize these latent variables in a structured manner.

3.3.2 Gaussian Membership Computation and Rule Activation

For the k-th expert, the model maintains a set of learnable Gaussian membership function parameters $\{\boldsymbol{\mu}_k^c, \boldsymbol{\sigma}_k\}_{k=1}^K$, where $\boldsymbol{\mu}_k^c \in \mathbb{R}^M$ is the center (ideal value) for each fuzzy variable, and $\boldsymbol{\sigma}_k = \exp(\boldsymbol{\rho}_k) + \sigma_{\min} \in \mathbb{R}^M$ is the standard deviation (condition looseness), with $\boldsymbol{\rho}_k$ being learnable log-standard-deviation parameters and $\sigma_{\min}$ a minimum standard deviation constant.

The membership degree of the current sample's fuzzy variables $\mathbf{z}$ for the k-th expert's m-th condition is computed using a Gaussian function:

$$u_{k,m} = \exp\left(-\frac{1}{2}\left(\frac{z_m - \mu_{k,m}^c}{\sigma_{k,m}}\right)^2\right). \tag{5}$$

The value $u_{k,m} \in (0, 1]$ quantifies "the degree to which the current sample satisfies the m-th premise condition of the k-th expert".

Then, the firing strength of the k-th rule is obtained by aggregating the memberships of all conditions via a product T-norm:

$$r_k = \prod_{m=1}^{M} u_{k,m}. \tag{6}$$

The physical meaning of r_k is the overall confidence that the current sample simultaneously satisfies all premise conditions of the k-th expert. The product T-norm embodies the "AND" operation in fuzzy logic, all conditions must be satisfied simultaneously to obtain a high firing strength.

3.3.3 Temperature-Scaled Softmax Gating

Unlike existing MoE methods that apply Softmax directly to meaningless linear projections, we apply temperature-scaled Softmax normalization to the rule firing strengths, which carry explicit physical semantics:

$$w_k = \frac{\exp(r_k/\tau)}{\sum_{j=1}^K \exp(r_j/\tau)}, \tag{7}$$

where τ is the temperature parameter that controls the sharpness of the gating distribution. The key advantage of this design is that the inputs to Softmax are physically meaningful rule firing strengths rather than arbitrary linear projection values, making the generation process of gating weights traceable.

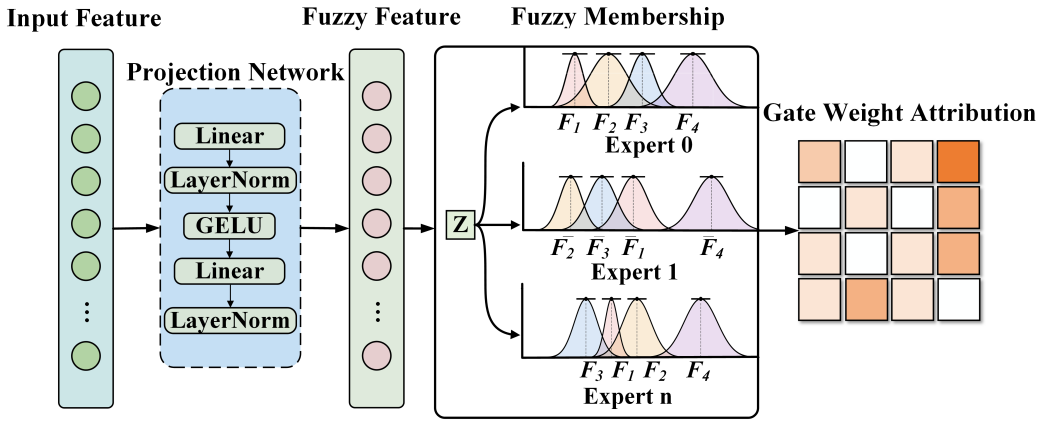


Figure 3: Detail of our Fuzzy Gate Architecture, which provides an inspectable basis for rule generation that aligns with intuitive regime-switching behaviors.

3.3.4 Dual-View Fusion

The gating computation is performed independently for each variable channel:

$$\mathbf{W}_{\text{conv}}^{(i)} = \mathcal{G}_{\text{conv}}(\mathbf{H}_{\text{conv}}^{(i)}) \in \mathbb{R}^{B\times K}, \quad \mathbf{W}_{\text{global}}^{(i)} = \mathcal{G}_{\text{global}}(\mathbf{H}_{\text{global}}^{(i)}) \in \mathbb{R}^{B\times K}. \tag{8}$$

The two gating outputs are averaged across channels and renormalized:

$$\tilde{\mathbf{W}} = \frac{1}{2}(\mathbf{W}_{\text{conv}} + \mathbf{W}_{\text{global}}), \quad \mathbf{W} = \frac{\tilde{\mathbf{W}}}{\sum_k \tilde{\mathbf{W}}_k + \epsilon} \in \mathbb{R}^{B\times C\times K}. \tag{9}$$

The fused $\mathbf{W}$ generates expert assignment weights independently for each variable channel at the sample-channel granularity, allowing different variables within the same sequence to select different forecasting mechanisms.

3.4 Expert Networks and Output Fusion

3.4.1 Expert Network Design

Each expert network $\mathcal{E}_k$ adopts a structure with an adaptive residual connection. The trunk network is a two-layer linear mapping: $\mathbb{R}^S \to \mathbb{R}^S \to \mathbb{R}^T$, where S is the number of input segments and T is the number of output segments. The shortcut branch is a single linear layer $\mathbb{R}^S \to \mathbb{R}^T$. The final output is a weighted combination of the two branches:

$$\mathcal{E}_k(\mathbf{X}_{\text{seg}}) = \alpha_{\text{short}} \cdot \text{Shortcut}(\mathbf{X}_{\text{seg}}) + \alpha_{\text{trunk}} \cdot \text{Trunk}(\mathbf{X}_{\text{seg}}), \tag{10}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^2$ are learnable fusion weights.

3.4.2 Weighted Fusion and Output Recovery

Each expert maps the historical segments $\mathbf{X}_{\text{seg}} \in \mathbb{R}^{B\times C\times T_{\text{seg}}\times S}$ to predicted segments $\hat{\mathbf{Y}}_k \in \mathbb{R}^{B\times C\times T_{\text{seg}}\times T}$. All expert outputs are aggregated via weighted summation using the gating weights:

$$\hat{\mathbf{Y}} = \sum_{k=1}^{K} \mathbf{W}_k \odot \hat{\mathbf{Y}}_k \in \mathbb{R}^{B\times C\times T_{\text{seg}}\times T}, \tag{11}$$

where $\odot$ denotes broadcast multiplication. The final forecasting is flattened to $\hat{\mathbf{Y}} \in \mathbb{R}^{B\times C\times P}$ (with $P = T_{\text{seg}} \times T$), denormalized, and transposed to $\mathbb{R}^{B\times P\times C}$ as the final output.

Models		Ours		WPMixer [AAAI2025]		SDE [SIGKDD2025]		TFPS [NIPS2025]		iTransformer [ICLR2024]		Time-MoE [ICLR2025]		PatchTST [ICLR2023]		Dlinear [AAAI2023]	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.362	0.383	0.374	0.387	0.387	0.402	0.398	0.413	0.394	0.409	**0.352**	**0.380**	0.376	0.397	0.396	0.410
	192	0.399	**0.406**	0.428	0.414	0.443	0.432	0.423	0.423	0.448	0.441	**0.389**	0.410	0.426	0.432	0.445	0.440
	336	**0.423**	**0.421**	0.462	0.437	0.492	0.457	0.484	0.461	0.492	0.465	0.424	0.439	0.469	0.457	0.487	0.465
	720	**0.433**	**0.449**	0.482	0.466	0.504	0.484	0.488	0.476	0.521	0.504	0.444	0.465	0.518	0.504	0.512	0.510
ETTh2	96	**0.271**	0.334	0.277	**0.330**	0.296	0.344	0.313	0.355	0.300	0.349	0.300	0.355	0.308	0.359	0.341	0.395
	192	**0.332**	**0.375**	0.351	0.377	0.381	0.395	0.405	0.410	0.381	0.399	0.354	0.383	0.380	0.406	0.481	0.479
	336	**0.321**	**0.380**	0.363	0.394	0.429	0.433	0.392	0.415	0.423	0.432	0.405	0.420	0.412	0.429	0.592	0.542
	720	**0.371**	**0.420**	0.405	0.427	0.435	0.444	0.410	0.433	0.426	0.445	0.465	0.466	0.435	0.456	0.840	0.661
ETTm1	96	**0.294**	**0.343**	0.334	0.368	0.322	0.363	0.327	0.367	0.341	0.376	0.309	0.355	0.323	0.364	0.345	0.373
	192	**0.335**	**0.367**	0.358	0.375	0.361	0.385	0.374	0.395	0.380	0.394	0.336	0.376	0.371	0.391	0.381	0.391
	336	**0.362**	**0.384**	0.384	0.397	0.401	0.414	0.401	0.408	0.419	0.418	0.374	0.405	0.398	0.408	0.415	0.415
	720	**0.426**	**0.416**	0.456	0.445	0.452	0.443	0.479	0.456	0.486	0.455	0.483	0.481	0.457	0.444	0.472	0.450
ETTm2	96	**0.166**	0.256	0.170	**0.251**	0.177	0.263	0.170	0.255	0.183	0.266	0.199	0.288	0.184	0.267	0.193	0.292
	192	**0.221**	**0.292**	0.235	0.295	0.248	0.311	0.235	0.296	0.252	0.312	0.248	0.322	0.246	0.304	0.284	0.361
	336	**0.276**	**0.328**	0.300	0.336	0.313	0.353	0.297	0.335	0.314	0.351	0.318	0.365	0.311	0.348	0.384	0.429
	720	**0.368**	**0.383**	0.391	0.392	0.418	0.415	0.401	0.397	0.411	0.406	0.465	0.451	0.418	0.414	0.556	0.523
Weather	96	**0.147**	**0.198**	0.163	0.205	0.165	0.213	0.154	0.202	0.175	0.215	0.158	0.212	0.175	0.217	0.196	0.256
	192	**0.194**	**0.242**	0.207	0.245	0.214	0.255	0.205	0.249	0.225	0.257	0.211	0.260	0.220	0.255	0.238	0.299
	336	**0.248**	**0.284**	0.267	0.291	0.273	0.297	0.262	0.289	0.279	0.298	0.273	0.307	0.279	0.297	0.281	0.330
	720	**0.315**	**0.336**	0.338	0.337	0.353	0.352	0.344	0.342	0.361	0.350	0.412	0.400	0.356	0.348	0.381	0.381
Electricity	96	**0.131**	**0.225**	0.166	0.260	0.147	0.245	0.149	0.236	0.148	0.240	-	-	0.180	0.272	0.210	0.301
	192	**0.145**	**0.241**	0.175	0.261	0.161	0.257	0.162	0.253	0.164	0.256	-	-	0.187	0.279	0.210	0.304
	336	**0.164**	**0.260**	0.193	0.282	0.176	0.274	0.200	0.310	0.177	0.270	-	-	0.204	0.295	0.223	0.319
	720	**0.203**	**0.294**	0.233	0.314	0.207	0.304	0.220	0.320	0.228	0.313	-	-	0.245	0.328	0.257	0.349

Table 1: Comparison of forecasting performance. The best results are highlighted in **bold** and the second-best results are underlined. While "-" denotes missing experimental results of the compared method on the corresponding dataset. Overall, our method consistently achieves the best performance across forecasting horizons = {96,192,336,720}.

4 EXPERIMENTS

4.1 SETUP

This experiment uses Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the evaluation metrics for the models. To validate the forecasting accuracy of our mode, we selected six commonly used datasets: 4 ETT datasetes(ETTh1, ETTh2, ETTm1, ETTm2) Zhou et al. (2021), Weather and Electricity for experimentation. We selected seven well-regarded time series models in the field as our baselines. These include both Linear-based and Transformer-based models: WPMixer Murad et al. (2025), SDE Weng et al. (2025), TFPS Sun et al. (2025), iTransformer Weng et al. (2025), Time-MoE Shi et al. (2025), PatchTST Nie et al. (2023) and DLinear Zeng et al. (2023). For the implementation details, see **Appendix B & Appendix C**.

4.2 MAIN RESULTS

Table 1 compares the forecasting performance of different methods on 6 benchmark datasets. Taking the Electricity dataset with the 720-step long forecasting horizon as an illustrative case, we compare Fuzzy-MoE against the strong MLP-based baseline iTransformer. On this challenging setting, our model achieves a MSE of 0.203 and MAE of 0.294, while iTransformer yields MSE=0.228 and MAE=0.313. Quantitatively, Fuzzy-MoE cuts MSE error by **10.96%** and MAE error by **6.07%** relative to iTransformer. This substantial performance gap arises because iTransformer relies on a single shared backbone to model all multivariate variables, failing to isolate distinct dynamic regimes across power load channels; by contrast, our dual-view fuzzy routing assigns specialized experts to different variables via traceable IF-THEN rules, precisely fitting heterogeneous periodic and fluctuation patterns within electricity signals under non-stationary distribution shifts. The gains are particularly evident on challenging datasets such as Weather and Electricity, demonstrating the

Deletion Type	ETTh1	ETTh2	Weather
Top-1 activated rule	**0.012**	**0.003**	**0.044**
Top-2 activated rule	0.008	**0.003**	0.012
Lowest rule	0.001	0.000	0.002

τ		0.3		0.8		2		5	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm2	96	0.167	0.257	**0.166**	0.256	0.167	0.256	**0.166**	**0.255**
	192	0.223	0.294	**0.221**	0.293	**0.221**	0.293	**0.221**	**0.292**
	336	0.278	0.330	0.276	0.328	0.276	0.328	**0.275**	**0.327**
	720	0.371	0.386	0.369	0.384	0.368	0.383	**0.367**	**0.382**

Table 2: Ablation results. (**Left**) ΔMAE after deleting rules with different activation strengths on ETTh1, ETTh2 and Weather. (**Right**) Forecasting performance under different temperature hyper-parameter τ on ETTm2. Best results are highlighted in **bold**.

Figure 4 presents the t-SNE visualization of the feature representations learned by the local branch, global branch, and their fusion. Compared with the individual branches, the fused representations exhibit a more structured and discriminative distribution, indicating that local and global state information provides complementary temporal cues. This enhanced feature separability enables the Fuzzy Gate to perform more accurate expert routing based on latent temporal states, thereby promoting expert specialization and collaboration, which ultimately contributes to the superior forecasting performance of Fuzzy-MoE.

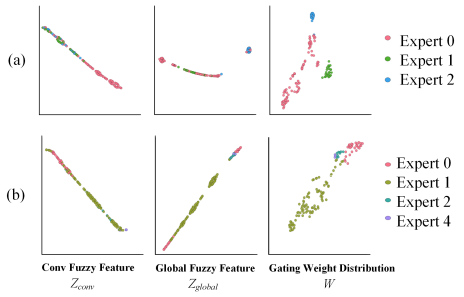


Figure 4: t-SNE Visualization of Local, Global, and Fused State Representations. (a) ETTh1; (b) ETTh2.

effectiveness of the proposed fuzzy-guided expert routing strategy in modeling complex temporal dynamics and improving forecasting accuracy.

4.3 Ablation Study

Table 2 (**Left**) presents the ΔMAE values resulting from removing rules with different activation strengths on the ETTh1, ETTh2 and Weather datasets, which intuitively reflects the contribution of fuzzy rules to forecasting performance.

Removing highly activated fuzzy rules leads to considerable MAE increases: $\Delta\text{MAE} = 0.012$ / 0.008 for top-1 / top-2 rule deletion on ETTh1, and $\Delta\text{MAE} = 0.003$ for both on ETTh2. Eliminating the least-activated rule produces negligible degradation ($\Delta\text{MAE} = 0.001$ for ETTh1, 0.000 for ETTh2), and similar behavior holds for Weather. This ablation demonstrates that high-confidence core fuzzy rules dominate routing and forecasting performance, while low-activation rules have marginal influence. It further validates that our Gaussian-membership-based routing can separate major/minor temporal patterns and adapt to latent dynamic regimes.

Additional ablation experiments including comparison between fuzzy gate and MLP gate are deferred to **Appendix D.5**.

5 Conclusion

This work proposes Fuzzy-MoE, an interpretable fuzzy Mixture-of-Experts framework for non-stationary multivariate forecasting. By dual-view Gaussian fuzzy routing, our model generates traceable IF-THEN rules and assigns experts per variable to resolve cross-channel heterogeneous dynamics. Sufficient benchmarks confirm our method achieves superior forecasting accuracy and intrinsic routing interpretability. For future work, we will extend the fuzzy rule extraction to online adaptive forecasting and integrate sparse expert activation to reduce computation overhead. In addition, the proposed framework offers a general, state-aware routing template that can be readily adapted to other regime-switching time series problems.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript language polishing, formula typesetting, and figure caption refinement. We did not use generative AI tools for the development of the core methodology, theoretical or mathematical claims, experimental design or implementation, or result analysis and interpretation; other research tasks requiring disclosure, such as synthetic data generation and qualitative data analysis, are not applicable to this work. Additionally, we used generative AI tools for preliminary literature-search assistance. We reviewed all AI-assisted work and manually cross-checked AI-polished text against our original arguments. All experiment-related descriptions were independently verified by two authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study evaluates FuzzyMoE on simulated, synthetic, and publicly available benchmark trajectories. No new data are collected from human participants, and no experiments involving human subjects or deployed decision-making systems are conducted. The datasets are used for model training, validation, and evaluation within the experimental protocols described in this paper.

REPRODUCIBILITY STATEMENT

Code and Implementation. Anonymous source code, pre-processed dataset files, configuration files, and running scripts are provided in the supplementary material archive. All experiments are implemented in PyTorch, with training conducted on a single NVIDIA RTX 3090 (24GB) GPU. Complete software dependency versions are listed in `requirements.txt` within the archive. Reproduction shell scripts are provided to launch experiments for all prediction horizons over each benchmark dataset.

Datasets. We use six public time-series forecasting benchmarks: ETTh1, ETTh2, ETTm1, ETTm2, Weather, and Electricity. Original public dataset sources are cited in the main paper. Train / validation / test splitting rules, normalization strategy, input sequence length, segmentation parameters are described in **Appendix B**. No custom filtering or manual data cleaning beyond standard preprocessing is applied.

Hyper-parameters and Experiment Protocols. All dataset-specific hyper-parameters (look-back length L, segment size T_{seg}, expert number K, convolution kernel/stride, learning rate, batch size, temperature τ) are summarized in **Appendix Table 4**. Random seed is fixed to 2025 for PyTorch, NumPy and CUDA to ensure deterministic behavior. Model selection uses the best validation metric checkpoint with early-stopping patience of 30. Ablation protocols for rule-deletion, temperature sweep, and MLP-gate replacement are documented in **Appendix C**. Evaluation metrics are MSE and MAE computed on held-out test sets.

REFERENCES

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P. Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen S. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, volume 162, pp. 5547–5569, 2022.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39, 2022.

Shanghua Gao, Teddy Koker, Owen Queen, Tom Hartvigsen, Theodoros Tsiligkaridis, and Marinka Zitnik. Units: A unified multi-task time series model. In *Advances in Neural Information Processing Systems*, 2024.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, 2023.

Matthias Hertel, Alexandra Nikoltchovska, Sebastian Pütz, Benjamin Schäfer, Ralf Mikut, and Veit Hagenmeyer. Explainable load forecasting with covariate-informed time series foundation models. In *Proceedings of the 17th ACM International Conference on Future and Sustainable Energy Systems*, pp. 612–626, 2026.

Qihe Huang, Zhengyang Zhou, Yangze Li, Kuo Yang, Binwu Wang, and Yang Wang. Many minds, one goal: Time series forecasting via sub-task specialization and inter-agent cooperation. In *Advances in Neural Information Processing Systems*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Yushan Jiang, Wenchao Yu, Geon Lee, Dongjin Song, Kijung Shin, Wei Cheng, Yanchi Liu, and Haifeng Chen. Timexl: Explainable multi-modal time series prediction with llm-in-the-loop. In *Advances in Neural Information Processing Systems*, 2025.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6555–6565, 2024.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International journal of forecasting*, 37(4): 1748–1764, 2021.

Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Junnan Li, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-moe: Empowering time series foundation models with sparse mixture of experts. In *Forty-second International Conference on Machine Learning*, volume 267, 2025.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Xiaowen Ma, Zhen-Liang Ni, Shuai Xiao, and Xinghao Chen. Timepro: Efficient multivariate long-term time series forecasting with variable- and time-aware hyper-state. In *Forty-second International Conference on Machine Learning*, volume 267, 2025.

Fares B. Mehouachi and Saif Eddin Jabari. Flowmixer: A depth-agnostic neural architecture for interpretable spatiotemporal forecasting. In *Advances in Neural Information Processing Systems*, 2025.

Md Mahmuddun Nabi Murad, Mehmet Aktukmak, and Yasin Yilmaz. Wpmixer: Efficient multi-resolution mixing for long-term time series forecasting. In *Thirty-Ninth AAAI Conference on Artificial Intelligence*, pp. 19581–19588, 2025.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Xiangfei Qiu, Xingjian Wu, Hanyin Cheng, Xvyuan Liu, Chenjuan Guo, Jilin Hu, and Bin Yang. Dbloss: Decomposition-based loss function for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 38, pp. 27741–27768, 2026.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

Yanru Sun, Zongxia Xie, Emadeldeen Eldele, Dongyue Chen, Qinghua Hu, and Min Wu. Learning pattern-specific experts for time series forecasting under patch-level distribution shift. In *Advances in Neural Information Processing Systems*, 2025.

Rui Wang, Renhao Xue, Ray Razi, Huan Song, and Hannah R Marlowe. Ame-ts: Anchored mixture-of-experts for time series forecasting. *arXiv preprint arXiv:2605.25166*, 2026.

Yang Wang, Jiaqi Zhu, Miaomiao Li, Jiang Liu, Yilin Li, Yi Yang, Jiafan Li, and Hongan Wang. Learning reliable and intuitive temporal logic rules for interpretable time series classification. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3067–3078, 2025.

Zixuan Weng, Jindong Han, Wenzhao Jiang, and Hao Liu. SDE: A simplified and disentangled dependency encoding framework for state space models in time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3168–3179, 2025.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, pp. 22419–22430, 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

Xingjian Wu, Xiangfei Qiu, Hanyin Cheng, Zhengyu Li, Jilin Hu, Chenjuan Guo, and Bin Yang. Enhancing time series forecasting through selective representation spaces: A patch perspective. In *Advances in Neural Information Processing Systems*, volume 38, pp. 23328–23354, 2026.

Xingyi Yang, Constantin Venhoff, Ashkan Khakzar, Christian Schröder de Witt, Puneet K. Dokania, Adel Bibi, and Philip Torr. Mixture of experts made intrinsically interpretable. In *Forty-second International Conference on Machine Learning*, volume 267, 2025.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, pp. 11121–11128, 2023.

Yunhe Zhang, Jinyu Cai, Zhihao Wu, Pengyang Wang, and See-Kiong Ng. Mixture of experts as representation learner for deep multi-view clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 22704–22713, 2025.

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 *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pp. 11106–11115, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, volume 162, pp. 27268–27286, 2022.

A APPENDIX

APPENDIX OVERVIEW

This appendix contains only supplementary material that is not fully expanded in the main paper. It provides routing gradients, code-level preprocessing details, omitted configuration values, and protocols for the auxiliary experiments. We do not repeat the main result table, the ablation result tables, or the visualization figures already shown in the main paper.

- Section A: routing gradients and numerical-stability details.
- Section B: dataset format, preprocessing, statistics, and train/validation/test splits.
- Section C: omitted training details and dataset-specific hyperparameters.
- Section D: protocols for rule deletion, temperature sweep, MLP-gate replacement, and visualization.

A. ADDITIONAL ROUTING DERIVATIVES

A.1 GAUSSIAN MEMBERSHIP GRADIENTS

Using the fuzzy-routing notation introduced in the main paper, let $\delta_{k,m} = z_m - \mu^c_{k,m}$. The gradient with respect to the learnable center parameter is

$$\frac{\partial u_{k,m}}{\partial \mu^c_{k,m}} = u_{k,m} \frac{\delta_{k,m}}{\sigma^2_{k,m}}. \tag{12}$$

Since $\partial \sigma_{k,m} / \partial \rho_{k,m} = \exp(\rho_{k,m})$, the gradient with respect to the raw scale parameter is

$$\frac{\partial u_{k,m}}{\partial \rho_{k,m}} = u_{k,m} \frac{\delta^2_{k,m} \exp(\rho_{k,m})}{\sigma^3_{k,m}}. \tag{13}$$

A.2 PRODUCT T-NORM GRADIENTS

For a single membership term $u_{k,t}$,

$$\frac{\partial r_k}{\partial u_{k,t}} = \prod_{\substack{m=1 \\ m \neq t}}^{M} u_{k,m} = \frac{r_k}{u_{k,t}}. \tag{14}$$

Combining Eq. 12, Eq. 13, and Eq. 14 gives the gradient chain from the forecasting loss to the fuzzy centers and scale parameters.

A.3 TEMPERATURE-SCALED SOFTMAX GRADIENTS

Let $\mathcal{Z} = \sum_{j=1}^{K} \exp(r_j / \tau)$. For any expert index t,

$$\frac{\partial w_k}{\partial r_t} = \frac{1}{\tau} w_k (\mathbf{1}_{k=t} - w_t). \tag{15}$$

Thus, in addition to its main-paper role in routing sharpness, τ also rescales gradients passing through the Softmax normalization by $1/\tau$.

A.4 NUMERICAL-STABILITY CHOICES

The implementation uses a minimum scale $\sigma_{\min} = 10^{-4}$ in the fuzzy membership function. It also adds a small ϵ when normalizing the averaged dual-view gate and when applying variance-based input normalization. These safeguards do not change the routing semantics, but they improve stability when fuzzy memberships become very small.

B. Dataset Format, Preprocessing, and Splits

B.1 Dataset Statistics

Table 3 summarizes the datasets used in the main experiments. These statistics are provided here because the main paper only names the benchmarks.

Table 3: Dataset statistics used by the forecasting experiments.

Dataset	Sampling Interval	Total Samples	Variables
ETTh1	1 hour	17420	7
ETTh2	1 hour	17420	7
ETTm1	15 min	69680	7
ETTm2	15 min	69680	7
Weather	10 min	52696	21
Electricity	1 hour	32304	321

B.2 Data Format

The code package expects each dataset to be stored as a NumPy archive:

`data/{dataset}.npz`

where `{dataset}` is one of `ETTh1`, `ETTh2`, `ETTm1`, `ETTm2`, `weather`, or `electricity`. Each array in the archive is loaded and stacked along the variable dimension. The resulting multivariate sequence is then used to construct sliding windows.

B.3 Normalization and Sliding Windows

For each dataset, the scaler is fitted only on the training portion. The full sequence is transformed with this training-fitted scaler, preventing validation/test statistics from leaking into normalization. A training sample is constructed as

$$(\mathbf{X}_{i:i+L}, \mathbf{X}_{i+L:i+L+P}), \tag{16}$$

where L is the input sequence length and P is the prediction horizon. Validation and test partitions include an overlap of length L at the beginning of the split so that the first validation/test target has enough historical context.

B.4 Split Ratios

The split ratios used in the code are dataset dependent:

Dataset Group	Train	Validation	Test
ETT datasets	60%	20%	20%
Weather/Electricity	70%	10%	20%

C. Omitted Training Details and Hyperparameters

C.1 Additional Training Details

The dimension of fuzzy variables is set to 3. SGD optimizer with initial learning rate 2e-5 is used for updating the parameters, the training epoch is 200, the dropout rate is 0.05, and the training batch size is set to 64. All experiments were implemented with PyTorch on a single NVIDIA 3090 24GB GPU. The following implementation details are not expanded in the main paper.

- Loss for optimization: mean squared error.

Table 4: Dataset-specific configurations used for the main experiments.

Dataset	Horizon	Config File	L	T_{seg}	K	k	s	Learning Rate
ETTh1	96	`ETTh1_96.toml`	336	24	3	8	4	5×10^{-5}
ETTh1	192	`ETTh1_192.toml`	336	24	3	4	4	2×10^{-4}
ETTh1	336	`ETTh1_336.toml`	336	24	3	8	4	8×10^{-5}
ETTh1	720	`ETTh1_720.toml`	336	24	3	24	4	2×10^{-5}
ETTh2	96	`ETTh2_96.toml`	720	4	5	4	4	2×10^{-5}
ETTh2	192	`ETTh2_192.toml`	720	4	5	4	4	5×10^{-5}
ETTh2	336	`ETTh2_336.toml`	720	4	5	12	4	2×10^{-5}
ETTh2	720	`ETTh2_720.toml`	720	4	5	12	4	2×10^{-5}
ETTm1	96	`ETTm1_96.toml`	336	2	8	4	4	5×10^{-5}
ETTm1	192	`ETTm1_192.toml`	336	2	4	8	4	8×10^{-4}
ETTm1	336	`ETTm1_336.toml`	336	2	8	24	4	5×10^{-5}
ETTm1	720	`ETTm1_720.toml`	336	2	4	24	4	2×10^{-5}
ETTm2	96	`ETTm2_96.toml`	336	2	2	4	4	2×10^{-5}
ETTm2	192	`ETTm2_192.toml`	336	2	4	8	4	2×10^{-5}
ETTm2	336	`ETTm2_336.toml`	336	2	4	4	4	2×10^{-5}
ETTm2	720	`ETTm2_720.toml`	336	2	2	4	4	2×10^{-5}
Weather	96	`weather_96.toml`	720	4	5	4	4	5×10^{-4}
Weather	192	`weather.toml`	720	4	5	4	2	8×10^{-4}
Weather	336	`weather.toml`	720	4	5	4	2	8×10^{-4}
Weather	720	`weather.toml`	720	4	5	4	2	8×10^{-4}
Electricity	96	`electricity.toml`	720	8	5	8	4	8×10^{-4}
Electricity	192	`electricity.toml`	720	8	5	8	4	8×10^{-4}
Electricity	336	`electricity.toml`	720	8	5	8	4	8×10^{-4}
Electricity	720	`electricity.toml`	720	8	5	8	4	8×10^{-4}

- Validation/test batch size: twice the configured training batch size.
- Optimizer and scheduler in the released training script: AdamW with StepLR, step size 20 and decay factor 0.75.
- Random seed: 2025 for PyTorch, CUDA, and NumPy; cuDNN benchmark disabled and deterministic mode enabled.
- Model selection: the checkpoint with the best validation criterion is saved. Early stopping uses patience 30.

C.2 Configuration Selection

For a dataset `D` and horizon `P`, the training script first searches for `configs/D_P.toml`. If this file is absent, it falls back to `configs/D.toml`. The command-line value of `pred_len` is then used as the effective prediction horizon. This is why a shared dataset-level file can be used for several horizons.

C.3 Dataset-Specific Hyperparameters

Table 4 lists the effective configuration files and the dataset-specific hyperparameters that are not fully listed in the main paper. Here L is the input sequence length, T_{seg} is the segment size, K is the number of experts, k is the convolution kernel size, and s is the convolution stride.

C.4 Software Environment

Version-pinned dependencies are provided in the supplementary `requirements.txt`. The implementation uses PyTorch together with NumPy, pandas, scikit-learn, einops, matplotlib, and tqdm.

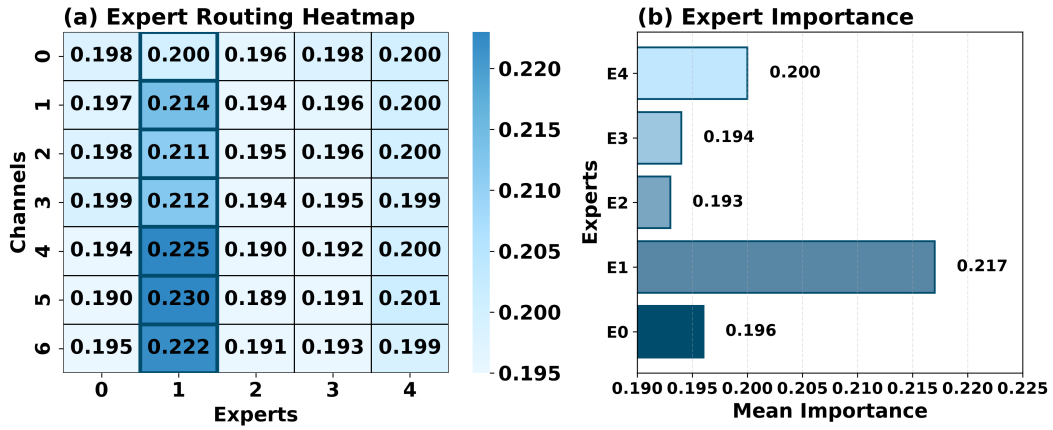


Figure 5: Visualization of expert weight attribution on the ETTh2 dataset. The routing weights are bounded by construction of the Gaussian memberships and remain balanced yet discriminative. We emphasize that this balance is a structural property of the bounded membership design rather than an explicit load-balancing mechanism; we further verify that no expert collapses via utilization diagnostics.

D. Auxiliary Experiment Protocols

D.1 Rule-Deletion Protocol

The main paper reports the error change after deleting high-activation and low-activation fuzzy rules. The protocol is:

1. Run the trained model on the evaluation split and record the fuzzy rule activations from both the convolutional and global gates.
2. Rank rules by their average activation.
3. For a selected rule, deactivate its corresponding Gaussian condition by moving its center far from the observed fuzzy region and shrinking its scale. In implementation, this is done by setting the selected center to a large value and the corresponding log-scale to a very small value.
4. Re-evaluate the model without retraining.
5. Report $\Delta\mathrm{MAE} = \mathrm{MAE}_{\mathrm{ablated}} - \mathrm{MAE}_{\mathrm{original}}$.

All other model weights, data splits, and evaluation settings are kept fixed.

D.2 Temperature-Sweep Protocol

The temperature sweep changes only the routing temperature reported in the main-paper temperature table. For each candidate, the model is trained and evaluated under the same dataset split, random seed, optimizer, and configuration except for the routing temperature. This isolates whether performance changes come from routing sharpness rather than from other hyperparameters.

D.3 MLP-Gate Replacement Protocol

The MLP-gate baseline replaces both fuzzy gates with MLP gates while preserving the remaining architecture: input normalization, segmentation, expert networks, output fusion, optimizer, scheduler, and data splits. This makes the comparison a direct test of the routing mechanism rather than a comparison between different forecasting backbones.

Models		Fuzzy Gate		MLP Gate	
Metric		MSE	MAE	MSE	MAE
ETTh2	96	**0.271**	**0.334**	0.303	0.362
	192	**0.332**	**0.375**	0.350	0.392
	336	**0.321**	**0.380**	0.331	0.389
	720	**0.371**	**0.420**	0.378	0.426
ETTm2	96	**0.166**	**0.256**	0.181	0.263
	192	**0.221**	**0.292**	0.232	0.299
	336	**0.276**	**0.328**	0.287	0.333
	720	**0.368**	**0.383**	0.376	0.389
Weather	96	**0.147**	**0.198**	0.149	0.204
	192	**0.194**	**0.242**	0.203	0.250
	336	**0.248**	**0.284**	0.252	0.290
	720	**0.315**	**0.336**	0.330	0.344

Table 5: Performance comparison of different gating mechanisms. The best results are highlighted in **bold**.

D.4 Routing Visualization Protocol

The routing heatmap is produced from the stored gate tensor after a forward pass. The average expert-weight view is computed by averaging gate weights over the selected evaluation batch and then plotting channel-by-expert weights. Figure 5 visualizes the average expert routing weights learned by Fuzzy-MoE on the ETTh2 dataset. Specifically, Expert 1 consistently receives higher routing weights across multiple channels, suggesting that it captures shared temporal dynamics, whereas the remaining experts focus on complementary latent patterns. These results demonstrate that the proposed Gaussian membership-based fuzzy routing enables adaptive and interpretable expert allocation, leading to more effective expert collaboration.

The t-SNE visualization uses the stored low-dimensional fuzzy projections from the convolutional branch, the global branch, and the final gate distribution. The dominant expert, defined by the largest gate weight, is used only as a visualization label.

D.5 Fuzzy-Gate versus MLP-Gate ablation results

Table 5 demonstrates that replacing the conventional MLP Gate with the proposed Fuzzy Gate consistently improves forecasting performance across different datasets and forecasting horizons. By introducing Gaussian fuzzy membership-based routing, the proposed gating mechanism provides a more discriminative and stable expert assignment than implicit MLP-based routing, leading to lower MAE and MSE in most settings. For example, when the horizon is set to 96 on ETTh2, compared with the standard MLP gate, our fuzzy gate achieves MSE of 0.271 and MAE of 0.334, corresponding to relative reductions of 10.56% and 7.73%, respectively. Consistent performance gains are further observed on ETTm2 and Weather.