

From Shared Demand Patterns to Local Uncertainty: Probabilistic Load Forecasting by Mixing Compact Adaptations

Haoran Li, *Member, IEEE*, Zhe Cheng, *Student Member, IEEE*, Yang Weng, *Senior Member, IEEE*,

Abstract—Probabilistic load forecasting has been widely studied for power-system operation and planning, but customer- and transformer-level forecasting introduces a distinct scalability challenge. At these levels, load uncertainty is strongly affected by customer behavior, weather, and mixed load composition, making it difficult for a single shared model to capture heterogeneous patterns. Using separate probabilistic models can improve local accuracy, but becomes costly to train, store, update, and validate at scale. To address this challenge, we develop a scalable customer-aware forecasting framework that learns common demand behavior through a shared model while adapting only a compact subset of parameters. Rather than using an independent model for each load or assigning each load to a specialized model, the proposed design learns a small bank of low-dimensional adaptation components and allows each load to combine them according to its forecasting characteristics. This preserves shared knowledge across customers while providing sufficient flexibility for heterogeneous and mixed load compositions. Experiments on 590 load profiles from the SMART-DS dataset show consistent improvements in deterministic accuracy and probabilistic quality over statistical, neural-network, Transformer-based, and pretrained time-series baselines, while retaining low storage and inference costs. The source code and data processing pipeline are available at [M²oE²-mLoRA](https://github.com/M2oE2-mLoRA).

Index Terms—Probabilistic load forecasting, distribution systems, uncertainty quantification, parameter-efficient learning, low-rank adaptation, mixture of experts.

I. INTRODUCTION

Load forecasting is a foundational function for modern power systems, providing essential information for economic dispatch [1], demand response [2], reliability assessment [3], event detection [4]–[6], and infrastructure planning [7]. As smart meters and advanced sensing infrastructures become increasingly available, forecasting practice is moving beyond aggregate system-level prediction toward fine-grained, bottom-up forecasting for individual customers in distribution feeders [8]–[10]. This shift is particularly important for distribution-system operation and can support distributed energy resource coordination [11], demand-side management [12], local market participation [13], and capacity planning [14]. Compared with point forecasting, probabilistic load forecasting further characterizes predictive uncertainty and provides richer information for risk-aware decision-making under variable and disaggregated demand [15].

Nevertheless, customer- and transformer-level probabilistic forecasting remains highly challenging in realistic distribution feeders with massive heterogeneous loads, including residential, commercial, industrial, and mixed transformer-level demand. Unlike more aggregated load forecasting, local power measurements reflect the combined effects of largely unobserved factors such as occupancy, business activity, appliance usage, electric-vehicle charging, distributed energy resources,

and customer composition behind transformers. These factors can produce strong differences in consumption and uncertainty patterns across customers, phases, locations, and aggregation levels, making it difficult for a single global forecaster to fully capture local behavior [16], [17]. At the same time, training and maintaining an independent deep probabilistic model for each customer or transformer is impractical for utilities, as storage, computation, and model-management costs grow rapidly with the number of loads. Therefore, scalable forecasting requires a mechanism that shares common knowledge across massive customers while adapting compactly to customer-specific demand and uncertainty characteristics.

Early approaches address local heterogeneity by building separate forecasting models for different customers, buildings, or load groups. Clustering-based methods partition customers according to load profiles and train separate predictors for the resulting clusters [18], [19]. Transfer-learning-based methods instead transfer knowledge learned from data-rich source loads, buildings, or customer groups to related target domains with limited data [20]–[22]. While clustering can substantially reduce complexity compared with maintaining an independent forecaster for every customer, storage and model-management costs may still become considerable when many clusters are required to represent heterogeneous loads. Transfer learning improves knowledge reuse, but often relies on source-to-target transfer and repeated adaptation across different loads.

The third category follows a shared-model adaptation strategy, where knowledge is learned jointly from many related loads and then adapted to specific customers or customer groups. Meta-learning methods learn a shared initialization or meta-model that can be quickly adapted to target household loads using limited local data [23], [24], but typically require an additional learning-to-adapt optimization procedure. Personalized federated learning trains a common model across consumers while allowing local parameter or gradient updates to preserve customer-specific characteristics [25]–[27]. Hypernetwork-based methods instead generate customer-specific model parameters or compact sub-parameters conditioned on customer information [10], [28], while fine-tuning approaches adapt only selected parameters of a pretrained model for different customer types [29]. These approaches substantially improve scalability by sharing most model knowledge across loads, but still face a capacity-efficiency tradeoff: the adapted parameter space must remain compact for large-scale deployment while being expressive enough to capture heterogeneous customer behavior and mixed transformer-level load compositions.

To address the capacity-efficiency challenge, we first investigate where customer-specific adaptation should be introduced

within a shared probabilistic forecaster. A key observation from our parameter-selection and ablation tests is that the forecasting model can be naturally separated into two functional stages: learning generalizable temporal and contextual representations across customers, and adapting these shared representations to customer-specific demand and uncertainty patterns. While the former can be largely shared, different customers may require different mappings from the common representation to their predictive distributions. This suggests that customer-specific adaptation should be concentrated at the post-fusion prediction stage, identifying the probabilistic output head as the key parameter subset for adaptation.

We therefore introduce low-rank adaptation (LoRA) [30] into the output head, freezing the shared parameters while learning compact low-rank updates for different load groups. This preserves the common output-mapping structure while substantially reducing the number of adapted parameters. However, a single LoRA update restricts adaptation and may not provide sufficient flexibility for heterogeneous customers and mixed transformer-level loads; simply increasing its rank improves capacity but also increases parameter cost. We thus propose a mixture-of-LoRA mechanism that represents each adaptation update as a weighted combination of multiple low-rank components, allowing diverse adaptation patterns to be composed from a small reusable parameter bank. The shared probabilistic forecaster is implemented using M²oE² [10], which incorporates external information as meta-knowledge and has demonstrated strong performance on benchmarks including GEFCom2014 [31].

The resulting model is termed M²oE²-mLoRA. We evaluate it on 590 residential, commercial, and industrial load profiles from the SMART-DS dataset [32] against statistical models, recurrent and convolutional neural networks, Transformers, and pretrained time-series foundation models. The evaluation covers deterministic accuracy, probabilistic forecasting quality, inference time, and storage cost. Results show that M²oE²-mLoRA improves both deterministic and probabilistic forecasting performance while maintaining low computational and storage overhead, demonstrating a favorable accuracy–uncertainty–cost tradeoff for customer- and transformer-level forecasting.

The rest of the paper is as follows. Section II formulates the problem. Section III provides preliminaries of M²oE². Section IV illustrates M²oE²-mLoRA framework. Section V presents numerical results, and Section VI concludes the paper.

II. PROBLEM FORMULATION

We define the customer-aware probabilistic load forecasting problem for massive heterogeneous feeder loads as follows.

- Goal: Learn a scalable probabilistic forecasting model that shares common knowledge across massive customer- and transformer-level loads, while adapting compactly to different load groups in realistic distribution feeders.
- Given: A customer/transformer index set $\mathcal{C}$, where each $c \in \mathcal{C}$ denotes one customer-level or transformer-level load. The historical load measurements of c are denoted as $\{x_{c,i}\}_{i=1}^N$, where N is the number of measurement points. Optional external measurements are denoted as

$\{w_{j,i} \mid j = 1, \dots, M; i = 1, \dots, N\}$, where M is the number of external data sources, such as weather, calendar variables, and electricity prices. The set $\mathcal{C}$ may contain K load groups, including residential, commercial, industrial, and transformer-level loads with different mixed customer compositions.

- Find: A shared M²oE² probabilistic forecaster f_Θ and a set of adaptation parameters $\{\theta_k\}_{k=1}^K$, where Θ denotes the shared global model parameters and θ_k denotes the LoRA parameters associated with the k -th load cluster.

III. PRELIMINARY: M²oE² GLOBAL FORECASTER

This section briefly reviews our previous M²oE² probabilistic forecaster [10], which serves as the shared global forecasting backbone f_Θ in this paper. The model is applicable to each customer or transformer $c \in \mathcal{C}$ and is trained using load data from all $c \in \mathcal{C}$. For notational simplicity, we omit the customer/transformer subscript c in this section. M²oE² builds on a probabilistic sequence base model and further treats external data as meta-knowledge to guide feature extraction. It uses MetaNets to transform heterogeneous external sources into candidate experts, a sparse gating network to select informative experts, and a residual anchor to preserve the no-external-data case. For more details, please review [10].

A. Variational Sequence As A Base Model

Let $\{x_i\}_{i=1}^N$ denote a load measurement sequence, where x_i is a scalar load measurement. A sequence base model, such as RNN, GRU, or LSTM, first encodes historical load information into a hidden state $\mathbf{h}_i$. For example, an RNN-type base model updates the hidden state as

$$\mathbf{h}_i = \rho(W_{hx}x_i + W_{hh}\mathbf{h}_{i-1} + \mathbf{b}_h), \quad (1)$$

where $\rho(\cdot)$ is the activation function, and W_{hx} , W_{hh} , and $\mathbf{b}_h$ are trainable parameters. The hidden state $\mathbf{h}_i$ summarizes historical load patterns and provides the temporal feature for future forecasting.

To quantify uncertainty, M²oE² adopts a variational probabilistic forecasting structure on top of the sequence base model. A latent vector $\mathbf{z}_{i+1}$ is introduced to represent stochastic factors that affect the future load. The encoder approximates the posterior distribution of $\mathbf{z}_{i+1}$ as

$$q_\Theta(\mathbf{z}_{i+1} \mid x_i, \mathbf{h}_{i-1}) = \mathcal{N}(\boldsymbol{\mu}_z(x_i, \mathbf{h}_{i-1}), \text{diag}(\boldsymbol{\sigma}_z^2(x_i, \mathbf{h}_{i-1}))). \quad (2)$$

where $\boldsymbol{\mu}_z(\cdot)$ and $\boldsymbol{\sigma}_z^2(\cdot)$ are neural-network mappings. The decoder then predicts the mean and variance of the future load distribution:

$$p_\Theta(x_{i+1} \mid \mathbf{z}_{i+1}) = \mathcal{N}(\mu_x(\mathbf{z}_{i+1}), \sigma_x^2(\mathbf{z}_{i+1})). \quad (3)$$

The model is trained by minimizing the negative evidence lower bound:

$$\mathcal{L}_{\text{ELBO}}(\Theta) = -\mathbb{E}_{q_\Theta(\mathbf{z}_{i+1} \mid x_i, \mathbf{h}_{i-1})} [\log p_\Theta(x_{i+1} \mid \mathbf{z}_{i+1})] + \lambda \text{KL}(q_\Theta(\mathbf{z}_{i+1} \mid x_i, \mathbf{h}_{i-1}) \parallel p(\mathbf{z}_{i+1})), \quad (4)$$

where $p(\mathbf{z}_{i+1}) = \mathcal{N}(\mathbf{0}, I)$, $\text{KL}(\cdot)$ is the Kullback–Leibler (KL) divergence-based regularization, and λ balances the likelihood term and latent-space regularization.

In general, the variational sequence base model maps historical load information into a conditional predictive distribution for future loads. Depending on the forecasting requirement, the decoder can directly provide distributional parameters, such as the predictive mean and variance, or generate Monte Carlo samples by repeatedly drawing z_{i+1} and decoding the corresponding load realizations. These samples can further be used to estimate other probabilistic statistics, including quantiles and prediction intervals, making the base model flexible for both point and probabilistic forecasting. Such a model serves as the base model for M²oE², as shown in the GRU encoder and mean/variance head blocks in Fig. 1.

B. External Meta-Knowledge through MetaNet and Gating

After establishing the variational sequence base model, the next step is to incorporate external data in a way that helps the model adapt to changing environmental and contextual factors. Instead of simply concatenating external variables with load inputs, M²oE² treats them as meta-knowledge that dynamically guides how the base model extracts load features.

For each time index i , optional external measurements are denoted as $\{w_{j,i} \mid j = 1, \dots, M\}$, where M is the number of external data sources, such as weather, calendar variables, and electricity prices. Thus, we introduce the following modules to construct the final meta representation ϕ_i in Eq. 6.

Module 1. MetaNet for building external experts. For the j -th external source, a MetaNet $g_j(\cdot)$ transforms $w_{j,i}$ into a candidate meta-representation:

$$\psi_{j,i} = \text{LN}(g_j(w_{j,i})), \quad j = 1, \dots, M, \quad (5)$$

where $\text{LN}(\cdot)$ denotes layer normalization. The layer normalization reduces scale and distributional heterogeneity among different external sources, so that each $\psi_{j,i}$ can be treated as a candidate expert for guiding the forecasting model.

Module 2. Gating network for selecting informative experts. Because the usefulness of external sources changes over time, M²oE² employs a gating network $l(\cdot)$ to automatically select informative experts according to the historical hidden state. The combined meta-representation is

$$\phi_i = \sum_{j=1}^M l_{j,i} \psi_{j,i} + \phi_0, \quad (6)$$

where $l_{j,i} = l(\mathbf{h}_{i-1})[j]$ is the gating weight for the j -th external expert and ϕ_0 is a trainable residual anchor. A top- m sparse softmax is applied to activate only the most relevant experts:

$$l_{j,i} = \frac{\exp(\tilde{l}_{j,i}) \mathbb{I}(j \in \mathcal{T}_i)}{\sum_{r \in \mathcal{T}_i} \exp(\tilde{l}_{r,i})}, \quad (7)$$

where $\tilde{l}_{j,i}$ is the pre-activation score, $\mathcal{T}_i$ contains the indices of the selected top- m experts, and $\mathbb{I}(\cdot)$ is the indicator function.

Module 3. Skip connection for preserving the no-external-data case. The residual anchor ϕ_0 allows M²oE² to fall back to the base forecasting structure when external information is uninformative. Specifically, if the selected external contribution vanishes, i.e., $\sum_{j=1}^M l_{j,i} \psi_{j,i} = \mathbf{0}$, then $\phi_i = \phi_0$. Therefore, the external-data module can be bypassed, preventing the

model from being worse than the corresponding no-external-data base model. The MetaNets, gate, and skip-connection are shown in the top-left part of Fig. 1.

C. Plug-and-Play Parameter Modulation

The meta-representation ϕ_i can be used to modulate different sub-parameters of the sequence base model in Subsection III-A, such as input-to-hidden W_{hx} , hidden-to-hidden W_{hh} , or output-layer parameters in μ_x and σ_x . In M²oE², a detailed analysis of parameter size, implementation cost, and adaptation effect shows that modulating the input-to-hidden matrix W_{hx} is an efficient choice: it allows external conditions to change how load inputs are encoded into temporal features while avoiding the large overhead of modulating recurrent or output parameters.

Directly replacing the internal matrix W_{hx} in Eq. (1), however, requires modifying the internal implementation of the sequence base model. To obtain a plug-and-play design, M²oE² uses an equivalent input transformation:

$$\mathbf{x}'_i = \phi_i x_i. \quad (8)$$

The transformed input $\mathbf{x}'_i$, as shown on the top left of Fig. 1, is then fed into the sequence base model. This formulation enables external meta-knowledge to adapt the feature extraction without changing the internal architecture of the base model. As a result, M²oE² can be integrated with different sequence base models, including RNN, GRU, and LSTM, and serves as an efficient global probabilistic forecasting backbone f_Θ . Here, $\Theta = \Theta_{\text{base}} \cup \Theta_{\text{meta}} \cup \Theta_{\text{gate}} \cup \{\phi_0\}$ includes the parameters of the variational sequence base model, the MetaNets $\{g_j(\cdot)\}_{j=1}^M$, the gating network $l(\cdot)$, and the residual anchor ϕ_0 .

In this paper, we further extend this backbone from external-data-aware forecasting to customer-aware forecasting by adapting sub-parameters $\{\theta_k\}_{k=1}^K$ for different load groups $k \in \{1, 2, \dots, K\}$, defined in Section II.

IV. PROPOSED MODEL

In this section, we propose M²oE²-mLoRA, as shown in Fig. 1. In the upper-left module (Stage 1), load history and external information are used to train a shared probabilistic forecaster; the pretrained backbone is then fixed during adaptation. In the lower-left module (Stage 2), load-profile features are clustered to identify representative customer and transformer patterns and determine the mixture weights used for adaptation. In the right module (Stage 3), these weights combine a shared bank of LoRA components to form customer-specific adaptations, which are applied to the probabilistic output heads for mean and variance prediction (Stage 4). This design reuses the same pretrained backbone across all loads while adapting only a compact set of parameters, avoiding full-model fine-tuning and supporting scalable deployment.

A. Load Clustering for Customer-Specific Fine-Tuning

Although the globally pre-trained M²oE² backbone captures common temporal and external-context knowledge across massive loads, it may not fully address the strong heterogeneity among residential, commercial, industrial, and transformer-level loads. Therefore, after global pre-training, customer-aware adaptation is required to further specialize the model

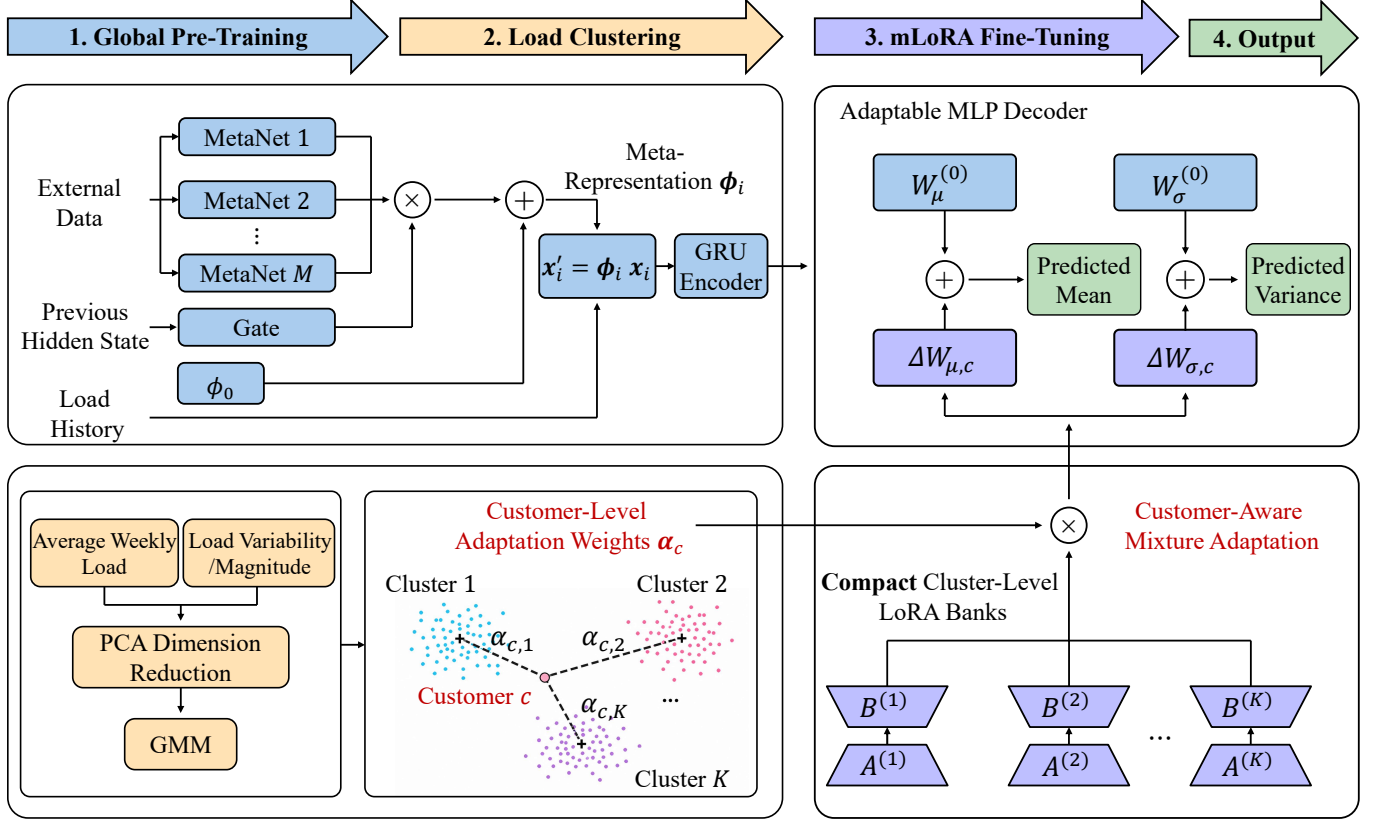


Fig. 1: M²oE²-mLoRA framework.

to different load behaviors by re-updating selected sub-parameters using specific groups of load data. Before performing such fine-tuning, it is necessary to understand how the load data are distributed across different behavioral patterns. To this end, we first conduct load clustering based on profile characteristics, so that loads with similar consumption patterns can provide structured guidance for the subsequent mixture-of-LoRA adaptation.

For each customer or transformer $c \in \mathcal{C}$, we use its historical training load data to construct one clustering representation. Specifically, the training data of load c are divided into Q weekly profiles, where each weekly profile is denoted by $x_{c,q} \in \mathbb{R}^S$ ($1 \leq q \leq Q$), and S is the number of time slots in one week. Since clustering is expected to capture the load-shape characteristics rather than the absolute load magnitude, each weekly profile is normalized by the maximum training-period load of customer c . Specifically, we have: $x_c^{\max} = \max_{1 \leq q \leq Q, 1 \leq s \leq S} x_{c,q,s}$, $\tilde{x}_{c,q} = \frac{x_{c,q}}{x_c^{\max} + \epsilon}$, where ϵ is a small constant for numerical stability.

Then, all normalized weekly profiles are averaged to obtain a representative weekly load shape p_c for each customer or transformer. To characterize load heterogeneity beyond the average shape, we further consider week-to-week variability and load magnitude. Specifically, we define $p_c = \frac{1}{Q} \sum_{q=1}^Q \tilde{x}_{c,q}$ and $v_{c,s} = \left[\frac{1}{Q} \sum_{q=1}^Q (\tilde{x}_{c,q,s} - p_{c,s})^2 \right]^{1/2}$. Here, p_c captures the average normalized weekly load shape, while $v_c = [v_{c,1}, \dots, v_{c,S}]^T$ characterizes the variability of each

time slot across weeks. To retain absolute demand information removed by normalization, we further define $a_c = \left[\frac{1}{QS} \sum_{q=1}^Q \sum_{s=1}^S x_{c,q,s}, x_c^{\max} \right]^T$, which contains the mean and maximum load magnitudes of the original profiles. The resulting feature vector therefore jointly characterizes typical load shape, temporal variability, and demand magnitude, corresponding to the three load-profile feature components in Fig. 1:

$$f_c = [p_c^T, v_c^T, a_c^T]^T. \quad (9)$$

Finally, the feature dimensions are standardized across the training loads and reduced using principal component analysis (PCA). The resulting low-dimensional representation u_c is used for subsequent clustering.

Given the reduced profile representations $\{u_c\}_{c \in \mathcal{C}}$, we further apply a Gaussian mixture model (GMM) to identify representative load groups. The use of GMM is motivated by the fact that load groups in realistic distribution feeders may not have clear hard boundaries. In particular, transformer-level loads can aggregate multiple types of customers, such as residential, commercial, and industrial loads, leading to mixed profile characteristics. In addition, customer behavior may vary under different weather conditions, calendar effects, and operating regimes. Therefore, instead of assigning each load to a single deterministic cluster, GMM provides soft membership weights that describe how strongly each customer or transformer is associated with different load clusters.

$$p(\mathbf{u}_c) = \sum_{k=1}^K \omega_k \mathcal{N}(\mathbf{u}_c | \mathbf{m}_k, \Sigma_k), \quad (10)$$

where K is the number of load groups, defined in Section II. ω_k is the mixture coefficient satisfying $\sum_{k=1}^K \omega_k = 1$, and $\mathbf{m}_k$ and Σ_k are the mean vector and covariance matrix of the k -th Gaussian component. The soft membership weight of load c with respect to cluster k is given by the posterior probability

$$\alpha_{c,k}^0 = \frac{\omega_k \mathcal{N}(\mathbf{u}_c | \mathbf{m}_k, \Sigma_k)}{\sum_{r=1}^K \omega_r \mathcal{N}(\mathbf{u}_c | \mathbf{m}_r, \Sigma_r)}, \quad k \in \{1, \dots, K\}. \quad (11)$$

The resulting vector $\alpha_c^0 = [\alpha_{c,1}^0, \alpha_{c,2}^0, \dots, \alpha_{c,K}^0]$ serves as the clustering-based soft membership assignment of customer c over the K load groups. This profile construction, dimensionality reduction, and soft-membership estimation correspond to the load-profile features, PCA, and GMM blocks in the lower-left part of Fig. 1.

B. Efficient Selection of Adapted Sub-Parameters

The clustering provides a structured description of load distributions across different customers. However, these clustering assignments only suggest which data subsets may require distinct customer-specific adaptation; they do not specify which sub-parameters of M²oE² should be adapted. A direct solution is to fine-tune the whole pre-trained model for each load cluster, but this would introduce large storage and computation overhead and weaken the scalability advantage of the shared backbone. Therefore, before constructing the mixture-of-LoRA adaptation, we first identify a small but effective subset of M²oE² parameters for customer-specific fine-tuning.

For customer-aware forecasting, different sub-parameters of M²oE² play different roles. The input transformation, external-data gating, and sequence feature extraction modules mainly learn common temporal and contextual representations shared by different loads. Directly adapting these modules for each customer group may disturb the globally learned representation and introduce unnecessary storage and computation overhead. In contrast, the **probabilistic output head** directly maps the learned latent representation to the predictive mean and variance, which are more sensitive to customer-specific load level, volatility, and uncertainty characteristics. Therefore, we select the output-head parameters as the adaptation target, so that each load group can adjust its predictive distribution while preserving the shared M²oE² backbone.

Following Eq. (3), for each customer or transformer $c \in \mathcal{C}$, the frozen M²oE² backbone first produces the latent representation $\mathbf{z}_{c,i+1}$. The probabilistic output head then maps $\mathbf{z}_{c,i+1}$ to the mean and variance of the future load distribution:

$$\begin{aligned} p_\Theta(x_{c,i+1} | \mathbf{z}_{c,i+1}) &= \mathcal{N}(\mu_x(\mathbf{z}_{c,i+1}), \sigma_x^2(\mathbf{z}_{c,i+1})), \\ \mu_x(\mathbf{z}_{c,i+1}) &= \rho\left(W_\mu^{(0)} \mathbf{z}_{c,i+1} + \mathbf{b}_\mu^{(0)}\right), \\ \log \sigma_x^2(\mathbf{z}_{c,i+1}) &= \rho\left(W_\sigma^{(0)} \mathbf{z}_{c,i+1} + \mathbf{b}_\sigma^{(0)}\right), \end{aligned} \quad (12)$$

where $W_\mu^{(0)}$ and $W_\sigma^{(0)}$ are the pre-trained output-head weight matrices for predicting the mean and variance, respectively, and $\mathbf{b}_\mu^{(0)}$ and $\mathbf{b}_\sigma^{(0)}$ are the corresponding bias terms. Here, the superscript (0) is added to $W_\mu^{(0)}$ and $W_\sigma^{(0)}$ to denote the

globally pre-trained output-head weights that remain frozen during customer-specific adaptation. We select $W_\mu^{(0)}$ and $W_\sigma^{(0)}$ as the target sub-parameters for customer-specific adaptation, as shown in the upper-right module of Fig. 1.

C. Soft-Membership-Guided Mixture-of-LoRA

Subsequently, we need to determine how these parameters should be adapted for different customers and transformers. A straightforward strategy is to learn one independent LoRA update for each load group. However, this hard assignment ignores the soft membership information obtained from GMM and may be insufficient for transformer-level loads with mixed load profiles. Therefore, in this subsection, we propose a soft-membership-guided mixture-of-LoRA mechanism.

The basis of the proposed mixture-of-LoRA mechanism is a single LoRA update for one load cluster. Therefore, we first define how LoRA adapts the selected output-head parameters for the k -th cluster. Instead of updating the full pre-trained matrices $W_\mu^{(0)}$ and $W_\sigma^{(0)}$, LoRA represents the cluster-specific updates by low-rank decompositions:

$$\begin{aligned} W_\mu^{(k)} &= W_\mu^{(0)} + \Delta W_\mu^{(k)}, & \Delta W_\mu^{(k)} &= B_\mu^{(k)} A_\mu^{(k)}, \\ W_\sigma^{(k)} &= W_\sigma^{(0)} + \Delta W_\sigma^{(k)}, & \Delta W_\sigma^{(k)} &= B_\sigma^{(k)} A_\sigma^{(k)}, \end{aligned} \quad (13)$$

where $A_\mu^{(k)} \in \mathbb{R}^{r \times d_z}$, $B_\mu^{(k)} \in \mathbb{R}^{d_x \times r}$, $A_\sigma^{(k)} \in \mathbb{R}^{r \times d_z}$, and $B_\sigma^{(k)} \in \mathbb{R}^{d_x \times r}$ are the trainable low-rank factors for the k -th cluster, with $r \leq \min(d_z, d_x)$. Here, d_z is the dimension of the latent representation $\mathbf{z}_{c,i+1}$, and d_x is the output dimension of the forecasted load. In multi-step forecasting, d_x is usually larger than 1 because the output head predicts a future horizon; for example, $d_x = 24$ for day-ahead hourly forecasting. During adaptation, the pre-trained matrices $W_\mu^{(0)}$ and $W_\sigma^{(0)}$ remain frozen, and only the low-rank factors are updated by gradients computed from the ELBO loss (see Eq. (4)) on the load data associated with the k -th cluster. Such a mixture of LoRA banks is presented on the bottom right of Fig. 1.

As discussed in Subsection IV-A, realistic load data often exhibit soft cluster memberships. Therefore, instead of selecting a single cluster-specific LoRA update, we construct a customer-specific adaptation as a linear mixture of multiple cluster-level LoRA updates. Let $\alpha_{c,k}$ denote the mixture weight of customer or transformer c on the k -th cluster, where $\alpha_{c,k} \geq 0$ and $\sum_{k=1}^K \alpha_{c,k} = 1$. The mixture-of-LoRA update is defined as:

$$\begin{aligned} \Delta W_{\mu,c} &= \sum_{k=1}^K \alpha_{c,k} \Delta W_\mu^{(k)} = \sum_{k=1}^K \alpha_{c,k} B_\mu^{(k)} A_\mu^{(k)}, \\ \Delta W_{\sigma,c} &= \sum_{k=1}^K \alpha_{c,k} \Delta W_\sigma^{(k)} = \sum_{k=1}^K \alpha_{c,k} B_\sigma^{(k)} A_\sigma^{(k)}. \end{aligned} \quad (14)$$

The adapted output-head matrices for customer or transformer c are then given by

$$\begin{aligned} W_{\mu,c} &= W_\mu^{(0)} + \Delta W_{\mu,c}, \\ W_{\sigma,c} &= W_\sigma^{(0)} + \Delta W_{\sigma,c}. \end{aligned} \quad (15)$$

Eq. (13) studies the LoRA update for a single load group, where the k -th cluster has its own adapted output-head matrices $W_\mu^{(k)}$ and $W_\sigma^{(k)}$. In contrast, Eq. (15) focuses on a

specific customer c , which may have soft memberships to multiple load clusters rather than belonging to only one group. Therefore, the customer-specific output heads $W_{\mu,c}$ and $W_{\sigma,c}$ are constructed by combining the LoRA updates from different clusters according to the membership weights $\{\alpha_{c,k}\}_{k=1}^K$.

The remaining question is how to determine the mixture weight $\alpha_{c,k}$ in Eq. (14). The GMM posterior $\alpha_{c,k}^0$ from Eq. (11) provides a useful intuition: for example, if customer or transformer c is very close to the first load cluster, then the first entry of α_c^0 will be close to 1 while the other entries will be close to 0. In this case, Eq. (14) should mainly activate the first LoRA update and suppress the others. Therefore, a direct option is to set $\alpha_{c,k} = \alpha_{c,k}^0$.

However, $\alpha_{c,k}^0$ is computed only from the load-profile clustering space, while the optimal LoRA contribution is determined by the forecasting loss in the parameter-adaptation space. These two spaces are related but not identical. Thus, directly fixing $\alpha_{c,k}$ as $\alpha_{c,k}^0$ may overly restrict the mixture. To preserve the clustering intuition while allowing forecasting loss-based adjustment, we treat α_c^0 as the initialization of α_c and introduce a trainable correction vector $\mathbf{r}_c = [r_{c,1}, \dots, r_{c,K}]$. Consequently, we can define:

$$\alpha_c = \text{softmax}(\log(\alpha_c^0 + \epsilon) + \tau \tanh(\mathbf{r}_c)), \quad (16)$$

where ϵ is a small constant for numerical stability, and $\tau \geq 0$ controls the maximum deviation from the GMM-based soft membership assignment. The log operator converts the initial membership vector α_c^0 from probability space to score space, so that applying softmax with $\mathbf{r}_c = \mathbf{0}$ recovers α_c^0 . Meanwhile, since $\tanh(\mathbf{r}_c)$ is bounded within $[-1, 1]$, the correction term cannot arbitrarily dominate $\log(\alpha_c^0 + \epsilon)$. Therefore, the mixture starts from the clustering-based membership assignment and remains guided by α_c^0 during training, while still allowing forecasting-loss-driven adjustment through the learnable correction vector $\mathbf{r}_c$.

We generate the correction score $\mathbf{r}_c$ using a shared lightweight neural network. For each customer-cluster pair (c, k) , the correction score is computed from the reduced customer profile $\mathbf{u}_c$ and the cluster center $\mathbf{m}_k$, defined in Eq. (11). Specifically, we have:

$$r_{c,k} = g_\Omega([\mathbf{u}_c; \mathbf{m}_k]), \quad k \in \{1, \dots, K\}, \quad (17)$$

where $g_\Omega(\cdot)$ is a small MLP shared by all clusters. The resulting correction vector is $\mathbf{r}_c = [r_{c,1}, \dots, r_{c,K}]$.

D. Training Algorithm and Online Prediction

In this subsection, we describe the offline training and online prediction procedures of the proposed M²oE²-mLoRA. The offline training procedure, summarized in Algorithm 1, consists of four stages: global pre-training, load clustering, cluster-wise LoRA pretraining, and joint mixture-of-LoRA adaptation. First, the M²oE² backbone is trained on all available loads by minimizing the ELBO objective in Eq. (4), and the pretrained backbone is then frozen. Second, feature preparation, PCA, and GMM are applied to obtain the clustering-based soft membership assignment α_c^0 . Third, each single-cluster LoRA basis in Eq. (13) is pretrained using the load data associated with its corresponding cluster. Since GMM provides soft

Algorithm 1 Offline Training of M²oE²-mLoRA

Require: Training load data $\{\{x_{c,i}\}_{i=1}^{N_c}\}_{c \in \mathcal{C}}$, external data $\{w_{j,i}\}_{j=1}^M$, number of clusters K , LoRA rank r , and correction bound τ .

Ensure: Frozen backbone $\Theta^{(0)}$, load features and statistics $\mathbf{f}_c$, GMM parameters $\{\omega_k, \mathbf{m}_k, \Sigma_k\}_{k=1}^K$, mixture-of-LoRA factors Φ_{LoRA} , and parameters Ω in $g_\Omega(\cdot)$.

- 1: **Step 1: Global pre-training**
 - 2: Train the M²oE² backbone using all loads by minimizing the ELBO objective in Eq. (4).
 - 3: Obtain the pretrained M²oE², i.e., $f_{\Theta^{(0)}}$, and freeze $\Theta^{(0)}$.
 - 4: **Step 2: Load clustering**
 - 5: **for** each customer or transformer $c \in \mathcal{C}$ **do**
 - 6: Construct the load-profile feature vector $\mathbf{f}_c$ by Eq. (9).
 - 7: **end for**
 - 8: Conduct PCA dimension reduction and obtain $\mathbf{u}_c$.
 - 9: Fit a GMM with K clusters on $\{\mathbf{u}_c\}_{c \in \mathcal{C}}$ using Eq. (10).
 - 10: Obtain the soft membership assignment α_c^0 by Eq. (11).
 - 11: **Step 3: Cluster-wise LoRA pretraining**
 - 12: **for** each load cluster $k = 1, \dots, K$ **do**
 - 13: Pretrain the single-cluster LoRA basis in Eq. (13) by minimizing the weighted ELBO loss in Eq. (18).
 - 14: Obtain the pretrained LoRA basis $\Phi_{\text{LoRA}}^{(k), \text{init}}$.
 - 15: **end for**
 - 16: Set $\Phi_{\text{LoRA}} \leftarrow \{\Phi_{\text{LoRA}}^{(k), \text{init}}\}_{k=1}^K$.
 - 17: **Step 4: Joint mixture-of-LoRA adaptation**
 - 18: **while** not converged **do**
 - 19: Sample a mini-batch of customer-time pairs $\mathcal{B}$.
 - 20: **for** each sampled customer or transformer c in $\mathcal{B}$ **do**
 - 21: Compute the correction vector $\mathbf{r}_c$ by Eq. (17).
 - 22: Compute the bounded corrected mixture weight α_c by Eq. (16).
 - 23: Construct $W_{\mu,c}$ and $W_{\sigma,c}$ by Eqs. (14) and (15).
 - 24: **end for**
 - 25: Obtain the predictive mean and variance by replacing $W_\mu^{(0)}$ and $W_\sigma^{(0)}$ in Eq. (12) with the customer-specific adapted matrices $W_{\mu,c}$ and $W_{\sigma,c}$.
 - 26: Compute the ELBO-based loss in Eq. (4) on the sampled mini-batch $\mathcal{B}$.
 - 27: Update only Φ_{LoRA} and Ω in $g_\Omega(\cdot)$ by backpropagation.
 - 28: **end while**
 - 29: **return** $\Theta^{(0)}$, $\{\omega_k, \mathbf{m}_k, \Sigma_k\}_{k=1}^K$, Φ_{LoRA} , and Ω .
-

memberships, the contribution of customer or transformer c to the k -th LoRA pretraining loss is weighted by $\alpha_{c,k}^0$:

$$\begin{aligned} \Phi_{\text{LoRA}}^{(k), \text{init}} = \arg \min_{\Phi_{\text{LoRA}}^{(k)}} & \sum_{c \in \mathcal{C}} \alpha_{c,k}^0 \\ & \times \sum_{i=1}^{N_c} \mathcal{L}_{\text{ELBO}}(x_{c,i+1}; W_\mu^{(k)}, W_\sigma^{(k)}), \end{aligned} \quad (18)$$

where $\Phi_{\text{LoRA}}^{(k)} = \{A_\mu^{(k)}, B_\mu^{(k)}, A_\sigma^{(k)}, B_\sigma^{(k)}\}$. The resulting pretrained single-cluster LoRA bases are then used to initialize the mixture-of-LoRA bank: $\Phi_{\text{LoRA}} \leftarrow \{\Phi_{\text{LoRA}}^{(k), \text{init}}\}_{k=1}^K$. This warm-start step gives each LoRA basis an explicit cluster-

TABLE I: Parameter-efficiency comparison of different customer-specific adaptation strategies.

Method	Required Parameters for Adaptation	Complexity	Scaling w.r.t. $ \mathcal{C} $
Full output-head adaptation	$\{W_{\mu,c}, W_{\sigma,c}\}_{c \in \mathcal{C}}$	$\mathcal{O}(2 \mathcal{C} d_x d_z)$	Linear in $ \mathcal{C} $
Independent LoRA per customer	$\{A_{\mu,c}, B_{\mu,c}, A_{\sigma,c}, B_{\sigma,c}\}_{c \in \mathcal{C}}$	$\mathcal{O}(2 \mathcal{C} r(d_x + d_z))$ $\mathcal{O}(2Kr(d_x + d_z))$ $+ \mathcal{O}(2d_u + 1)$ $+ \mathcal{O}(Sd_u + S)$	Linear in $ \mathcal{C} $
Proposed M ² oE ² -mLoRA	$\{A_{\mu}^{(k)}, B_{\mu}^{(k)}, A_{\sigma}^{(k)}, B_{\sigma}^{(k)}\}_{k=1}^K, g_{\Omega}, \text{PCA, GMM}$	$+ \mathcal{O}(Kd_u + \frac{Kd_u(d_u+1)}{2} + K)$	Independent of $ \mathcal{C} $

level meaning before the final joint adaptation stage. After the warm-start, the cluster-level LoRA bases and the correction network $g_{\Omega}(\cdot)$ are jointly optimized while the M²oE² backbone remains frozen. During joint adaptation, only Φ_{LoRA} and Ω are updated by minimizing the ELBO-based loss in Eq. (4), where the predictive mean and variance are computed using the customer-specific adapted output heads $W_{\mu,c}$ and $W_{\sigma,c}$.

During online prediction, the frozen M²oE² backbone is reused for all customers. For each customer or transformer c , the trained PCA, GMM, correction network, and mixture-of-LoRA factors are used to construct the customer-specific adapted output heads $W_{\mu,c}$ and $W_{\sigma,c}$. The final predictive mean and variance are then obtained by applying Eq. (12) with $W_{\mu}^{(0)}$ and $W_{\sigma}^{(0)}$ replaced by $W_{\mu,c}$ and $W_{\sigma,c}$.

E. Parameter Efficiency and Deployment Complexity

In realistic distribution feeders, $|\mathcal{C}|$ (the number of customers or transformers) can be very large because utilities may need to forecast thousands or even hundreds of thousands of customer- and transformer-level loads. Therefore, the storage complexity and scaling with respect to $|\mathcal{C}|$ becomes a critical deployment bottleneck. Table I presents the parameter-efficiency comparison among three customer-specific adaptation strategies. Here, d_z is the dimension of the latent representation, d_x is the output dimension of the forecasting horizon, r is the LoRA rank, K is the number of load clusters, S is the dimension of the normalized weekly profile, and d_u is the PCA-reduced profile dimension.

For full output-head adaptation, each customer requires its own mean and variance output matrices, resulting in $\mathcal{O}(2|\mathcal{C}|d_x d_z)$ additional parameters. This cost grows linearly with the number of loads and becomes expensive in large-scale feeders. Independent LoRA per customer reduces the cost by replacing each full matrix update with low-rank factors. For each customer, the mean and variance heads require two LoRA updates, leading to $\mathcal{O}(2|\mathcal{C}|r(d_x + d_z))$ additional parameters. Although this is much smaller than full output-head adaptation when $r \ll \min(d_x, d_z)$, the storage cost still scales linearly with $|\mathcal{C}|$, since each customer or transformer needs an independent adapter.

In contrast, the proposed M²oE²-mLoRA stores only K cluster-level LoRA updates shared by all customers, giving $\mathcal{O}(2Kr(d_x + d_z))$ parameters for the mean and variance heads. In addition, the lightweight (1 layer) neural network g_{Ω} maps $[u_c; \mathbf{m}_k] \in \mathbb{R}^{2d_u}$ to the scalar correction score $r_{c,k}$, requiring $\mathcal{O}(2d_u + 1)$ parameters. The clustering-related components introduce $\mathcal{O}(Sd_u + S)$ parameters for the PCA projection and mean profile, and $\mathcal{O}(Kd_u + \frac{Kd_u(d_u+1)}{2} + K)$ parameters for

the full-covariance GMM. Since these components are shared across all customers and $K \ll |\mathcal{C}|$, the proposed method avoids storing customer-specific adapters and keeps the additional parameter cost independent of the number of loads.

V. EXPERIMENTS

A. Setup

SMART-DS (Synthetic Models for Advanced, Realistic Testing: Distribution Systems and Scenarios) provides public distribution network models and load time series for several U.S. regions, including San Francisco, Greensboro, and Austin [32]. The models are designed to reproduce realistic statistical and operational characteristics of utility distribution systems, with active and reactive power profiles available at 15-minute intervals. We use 590 residential, commercial, and industrial load profiles from the 2018 dataset and aggregate active power to hourly resolution for forecasting. Selected profiles contain complete observations throughout 2018 and have a 95th-percentile hourly demand of at least 4 kW from January to October, spanning demand levels from 4 kW to above 128 kW.

Training uses January-October data, validation uses November, and testing uses December. Each day at midnight, the model forecasts the next 24 hours. Using complete weekly samples yields 30 forecasting times per profile from December 1–30, 2018, corresponding to 17,700 daily forecasts and 424,800 hourly predictions. Each sample spans two weeks. The model first encodes 168 hours of historical load, then updates this representation with newly available observations before each daily forecast and predicts the next 24 hours' load.

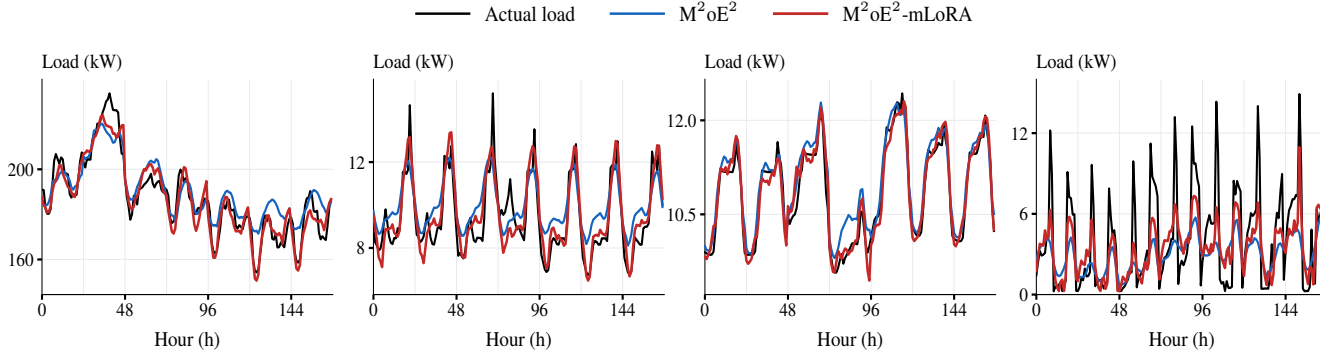
We evaluate forecasting performance using seven metrics [10], [33]: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Weighted Absolute Percentage Error (WAPE), Continuous Ranked Probability Score (CRPS), Quantile Score (QS), and Winkler Score (WS). MSE measures profile-normalized squared error; RMSE and MAE quantify prediction errors in kW; WAPE measures absolute error relative to aggregate demand; CRPS evaluates the overall predictive distribution; QS assesses the accuracy of predicted quantiles; and WS evaluates prediction intervals by jointly considering their coverage and width. Lower values indicate better performance for all metrics.

We compare M²oE²-mLoRA against statistical, recurrent/convolutional, Transformer-based, and pretrained time-series foundation models. The benchmarks include ARIMA + Fourier, GRU [34], LSTM [35], CNN-GRU, DeepAR [36], iTransformer [37], PatchTST [38], Chronos-2 [39], TimesFM-2.5 [40], [41], and TimesFM-2.5 + LoRA [30], covering

TABLE II: Forecasting performance and computational cost.

Model	MSE (10^{-3})	RMSE	MAE	WAPE	CRPS	QS	WS ₉₀	WS ₅₀	Time	Storage
ARIMA + Fourier	5.47	8.79	3.26	8.19	2.45	1.10	20.96	11.16	6.86*	0.18
CNN-GRU	3.92	6.58	2.37	5.95	1.75	0.79	15.52	7.79	0.04	0.33
TimesFM-2.5 zero shot	3.74	7.22	2.35	5.91	1.80	0.83	17.72	8.03	4.92	882.30
iTransformer	3.69	6.22	2.14	5.37	1.67	0.77	16.99	7.42	0.06	18.77
LSTM	3.51	4.90	1.77	4.45	1.31	0.59	11.88	5.84	0.10	0.79
GRU	3.49	5.44	1.98	4.97	1.48	0.68	14.26	6.57	0.06	0.60
PatchTST	3.35	5.43	1.87	4.68	1.45	0.68	15.36	6.44	0.04	2.32
Chronos-2 zero shot	3.21	6.16	2.02	5.07	1.48	0.67	12.59	6.61	10.15	455.77
DeepAR	3.18	5.08	1.71	4.28	1.45	0.72	12.78	6.98	11.62	0.77
TimesFM-2.5 + LoRA	3.15	5.32	1.73	4.35	1.37	0.64	14.99	6.07	4.93	882.30
M ² oE ²	2.98	4.87	1.79	4.49	1.32	0.59	11.29	5.90	0.08	0.40
M ² oE ² -mLoRA	2.72	4.50	1.60	4.16	1.18	0.48	11.01	5.54	0.11	0.58

MSE uses each customer’s training load range and is reported in units of 10^{-3} . RMSE, MAE, CRPS, QS, WS₉₀, and WS₅₀ are in kW, while WAPE is reported in percent. Time is reported in ms/forecast and storage in MiB. Bold values indicate the best result in each column before rounding. Accuracy is evaluated on the full December test set. Time is the median processing time divided by 4,130 daily forecasts from December 3–9, with each forecast covering 24 hours. Neural models are evaluated on an NVIDIA Tesla T4. *ARIMA is evaluated on an Intel Xeon CPU with two configured threads and is excluded from GPU speed comparisons. Storage includes the forecasting model and required static routing data, with one MiB equal to 2^{20} bytes.

Fig. 2: Examples of load forecasts from M²oE² and M²oE²-mLoRA.

deterministic and probabilistic forecasting as well as zero-shot and adapted foundation models. We further compare against an M²oE² ablation [10] without the proposed mLoRA adaptation, which serves as the shared base model for evaluating the contribution of customer-specific adapters.

B. Forecasting Performance and Computational Cost

Table II shows that M²oE²-mLoRA achieves the best performance across all evaluated deterministic and probabilistic forecasting metrics. Its MSE is 8.72% lower than that of M²oE² and 13.65% lower than that of TimesFM-2.5 with LoRA, demonstrating the benefit of cluster-specific mLoRA adaptation. The following figures further illustrate the forecasting behavior for representative load profiles.

Fig. 2 compares the actual loads with forecasts from M²oE² and M²oE²-mLoRA for representative profiles with different demand levels and temporal patterns. Each panel shows seven consecutive day-ahead forecasts. For the first two profiles, customer-aware adaptation reduces overestimation between daily peaks, while for the third profile, it more accurately captures the lower-load period around hours 72–90. The last profile exhibits more irregular fluctuations, for which M²oE²-mLoRA better tracks low-load intervals and more accurately recovers a sharp demand spike. Fig. 3 compares one-week forecasts for six test profiles across all benchmark models. Overall, M²oE²-mLoRA tracks the observed load levels and daily variations more closely than the compared methods.

TABLE III: Sensitivity to the number of clusters.

K	Adapter parameters	November MSE ($\times 10^{-3}$)
14	1,971,200	2.97
15	2,112,000	2.96
16	2,252,800	2.97

Inference cost is evaluated on the December 3–9 test period under the same forecasting setting for all methods. As shown in Table II, M²oE²-mLoRA requires only 0.11 ms per forecast and 0.58 MiB of storage, remaining substantially more efficient than large pretrained foundation models while adding only modest overhead over the shared M²oE² base. This demonstrates that the proposed cluster-specific adaptation improves forecasting accuracy without sacrificing computational scalability.

C. Parameter Sensitivity

We evaluate the sensitivity of M²oE²-mLoRA to the number of load clusters using the same 590 profiles and chronological data split. The clusters are constructed from training data based on similarities in training-period load-profile features, and each cluster is assigned its own mLoRA adapter. As K increases, the model can represent more diverse customer-specific adaptation patterns, but the number of adapter parameters and overall model size also increase approximately linearly. Table III compares $K = 14, 15, 16$; the lowest

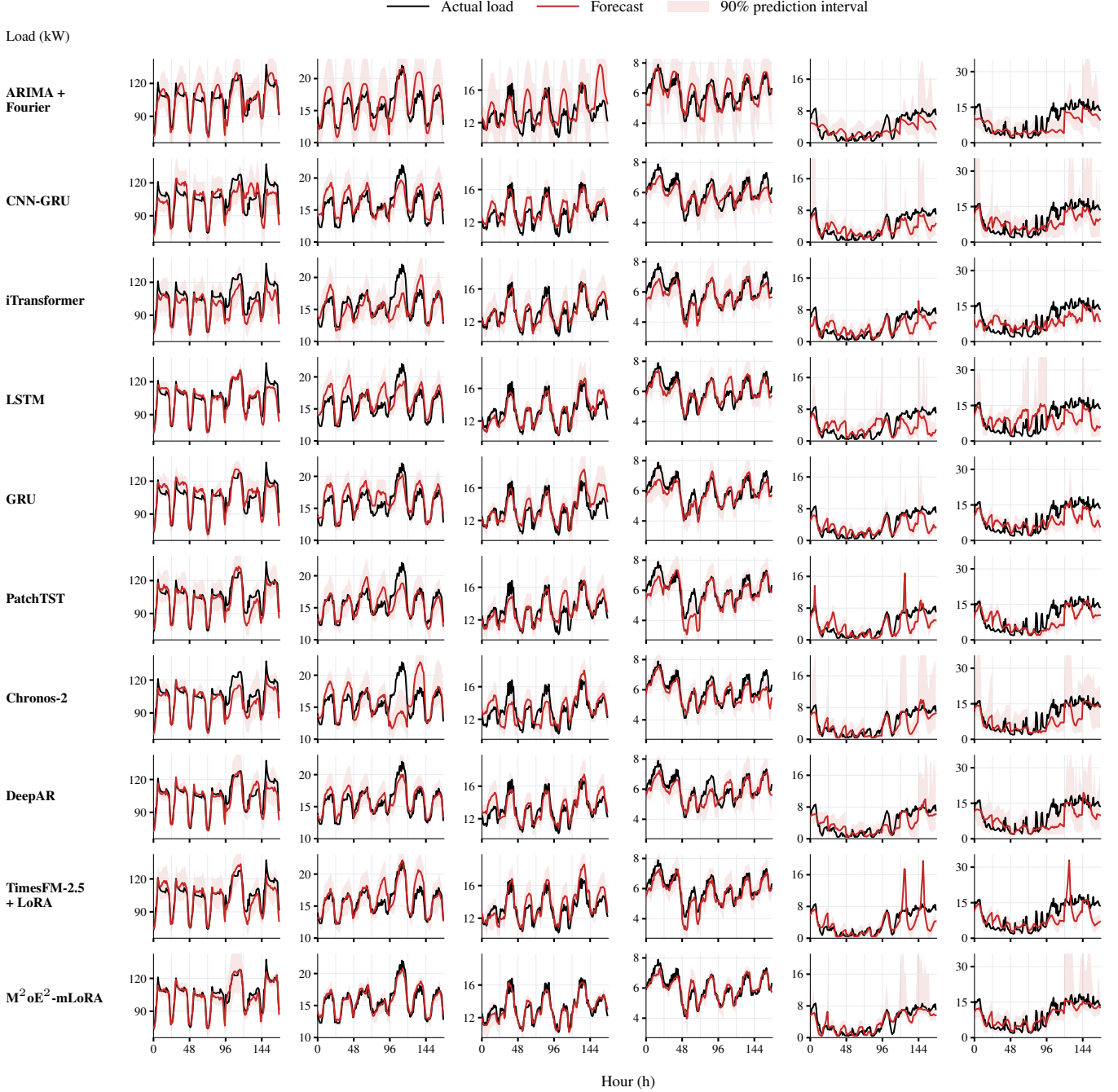


Fig. 3: Weekly load forecasts for six selected SMART-DS profiles. Rows show all nine benchmark configurations, and M^2oE^2 -mLoRA. Each column uses the same actual loads and vertical scale throughout.

November validation MSE is obtained with $K = 15$, while the overall performance remains stable across the tested values, supporting the use of 15 clusters in the final model.

VI. CONCLUSION

This paper studied scalable customer-aware probabilistic forecasting for heterogeneous customer- and transformer-level loads without training a separate full model for each load. The proposed M^2oE^2 -mLoRA framework identifies the probabilistic output head as the key parameter subset for customer-specific adaptation and introduces a soft-membership-guided mixture-of-LoRA mechanism, allowing each load to combine

compact cluster-level adaptations while preserving the shared forecasting backbone. Experiments on 590 load profiles from the SMART-DS dataset show that M^2oE^2 -mLoRA improves all evaluated deterministic and probabilistic forecasting metrics over the shared M^2oE^2 model and achieves the best overall forecasting performance among the evaluated methods, with low additional storage and inference cost. These results demonstrate that mixture-based parameter-efficient adaptation provides a scalable approach to capturing heterogeneous demand and uncertainty patterns in customer- and transformer-level probabilistic forecasting.

REFERENCES

- [1] J. Han, L. Yan, and Z. Li, "A task-based day-ahead load forecasting model for stochastic economic dispatch," *IEEE Transactions on Power Systems*, vol. 36, no. 6, pp. 5294–5304, 2021.
- [2] S. H. Pramono, M. Rohmatillah, E. Maulana, R. N. Hasanah, and F. Hario, "Deep learning-based short-term load forecasting for supporting demand response program in hybrid energy system," *Energies*, vol. 12, no. 17, p. 3359, 2019.
- [3] A. Saxena, R. Shankar, E. El-Saadany, M. Kumar, O. Al Zaabi, K. Al Hosani, and U. R. Muduli, "Intelligent load forecasting and renewable energy integration for enhanced grid reliability," *IEEE Transactions on Industry Applications*, 2024.
- [4] H. Li, H. Tong, and Y. Weng, "Domain adaptation in physical systems via graph kernel," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 868–876. [Online]. Available: <https://doi.org/10.1145/3534678.3539380>
- [5] H. Li, Z. Ma, and Y. Weng, "A transfer learning framework for power system event identification," *IEEE Transactions on Power Systems*, vol. 37, no. 6, pp. 4424–4435, 2022.
- [6] H. Li, Z. Ma, Y. Weng, E. Blasch, and S. Santoso, "Structural tensor learning for event identification with limited labels," *IEEE Transactions on Power Systems*, vol. 38, no. 6, pp. 5314–5328, 2023.
- [7] X. Zhang, T. K. Chau, Y. H. Chow, T. Fernando, and H. H.-C. Iu, "A novel sequence to sequence data modelling based cnn-lstm algorithm for three years ahead monthly peak load forecasting," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 1932–1947, 2023.
- [8] C. Ye, Y. Ding, P. Wang, and Z. Lin, "A data-driven bottom-up approach for spatial and temporal electric load forecasting," *IEEE Transactions on Power Systems*, vol. 34, no. 3, pp. 1966–1979, 2019.
- [9] H. Li, M. Guo, Y. Weng, M. Ilic, and G. Ruan, "Examnn: An environment-driven adaptive rnn for learning non-stationary power dynamics," *arXiv preprint arXiv:2505.17488*, 2025.
- [10] H. Li, M. Guo, M. Ilic, Y. Weng, and G. Ruan, "External data-enhanced meta-representation for adaptive probabilistic load forecasting," *IEEE Transactions on Smart Grid*, pp. 1–1, 2026.
- [11] A. Attarha, P. Scott, and S. Thiébaux, "Affinely adjustable robust admm for residential der coordination in distribution networks," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1620–1629, 2019.
- [12] X. Kong, C. Li, F. Zheng, and C. Wang, "Improved deep belief network for short-term load forecasting considering demand-side management," *IEEE transactions on power systems*, vol. 35, no. 2, pp. 1531–1538, 2019.
- [13] S. Schreck, I. P. de La Comble, S. Thiem, and S. Niessen, "A methodological framework to support load forecast error assessment in local energy markets," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3212–3220, 2020.
- [14] J. Ponočko and J. V. Milanović, "Forecasting demand flexibility of aggregated residential load using smart meter data," *IEEE Transactions on Power Systems*, vol. 33, no. 5, pp. 5446–5455, 2018.
- [15] B. Wang, M. Mazhari, and C. Chung, "A novel hybrid method for short-term probabilistic load forecasting in distribution networks," *IEEE Transactions on Smart Grid*, vol. 13, no. 5, pp. 3650–3661, 2022.
- [16] Y. Wang, N. Zhang, Y. Tan, T. Hong, D. S. Kirschen, and C. Kang, "Combining probabilistic load forecasts," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3664–3674, 2018.
- [17] H. Li, Y. Weng, V. Vittal, and E. Blasch, "Distribution grid topology and parameter estimation using deep-shallow neural network with physical consistency," *IEEE Transactions on Smart Grid*, vol. 15, no. 1, pp. 655–666, 2024.
- [18] T. K. Wijaya, M. Vasirani, S. Humeau, and K. Aberer, "Cluster-based aggregate forecasting for residential electricity demand using smart meter data," in *2015 IEEE international conference on Big data (Big data)*. IEEE, 2015, pp. 879–887.
- [19] F. Fahiman, S. M. Erfani, S. Rajasegarar, M. Palaniswami, and C. Leckie, "Improving load forecasting based on deep learning and k-shape clustering," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 4134–4141.
- [20] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. Capretz, "Transfer learning with seasonal and trend adjustment for cross-building energy forecasting," *Energy and Buildings*, vol. 165, pp. 352–363, 2018.
- [21] D. Wu and W. Lin, "Efficient residential electric load forecasting via transfer learning and graph neural networks," *IEEE Transactions on Smart Grid*, vol. 14, no. 3, pp. 2423–2431, 2022.
- [22] Y. Tian, L. Sehovac, and K. Grolinger, "Similarity-based chained transfer learning for energy forecasting with big data," *IEEE Access*, vol. 7, pp. 139 895–139 908, 2019.
- [23] Y. He, F. Luo, and G. Ranzi, "Transferrable model-agnostic meta-learning for short-term household load forecasting with limited training data," *IEEE Transactions on Power Systems*, vol. 37, no. 4, pp. 3177–3180, 2022.
- [24] G. Tsoumplekas, C. Athanasiadis, D. I. Doukas, A. Chrysopoulos, and P. Mitkas, "Few-shot load forecasting under data scarcity in smart grids: A meta-learning approach," *Energies*, vol. 18, no. 3, p. 742, 2025.
- [25] Y. Wang, N. Gao, and G. Hug, "Personalized federated learning for individual consumer load forecasting," *CSEE Journal of Power and Energy Systems*, vol. 9, no. 1, pp. 326–330, 2022.
- [26] C. Briggs, Z. Fan, and P. Andras, "Federated learning for short-term residential load forecasting," *IEEE open access journal of power and energy*, vol. 9, pp. 573–583, 2022.
- [27] Y. He, F. Luo, M. Sun, and G. Ranzi, "Privacy-preserving and hierarchically federated framework for short-term residential load forecasting," *IEEE Transactions on Smart Grid*, vol. 14, no. 6, pp. 4409–4423, 2023.
- [28] M. U. Danish and K. Grolinger, "Leveraging hypernetworks and learnable kernels for consumer energy forecasting across diverse consumer types," *IEEE Transactions on Power Delivery*, vol. 40, no. 1, pp. 75–87, 2024.
- [29] D. Gupta, A. Bhatti, S. Parmar, C. Dan, Y. Liu, B. Shen, and S. Lee, "Low-rank adaptation of time series foundational models for out-of-domain modality forecasting," in *Proceedings of the 26th International Conference on Multimodal Interaction*, 2024, pp. 382–386.
- [30] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models," *Iclr*, vol. 1, no. 2, p. 3, 2022.
- [31] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," pp. 896–913, 2016.
- [32] B. Palmintier *et al.*, "SMART-DS synthetic electrical network data OpenDSS models for SFO, GSO, and AUS," Open Energy Data Initiative, 2020. [Online]. Available: <https://data.openei.org/submissions/2981>
- [33] H. Li, Z. Cheng, M. Guo, Y. Weng, Y. Sun, V. Tran, and J. Chainaranont, "M2oe2-gl: A family of probabilistic load forecasters that scales to massive customers," in *2026 IEEE Power & Energy Society General Meeting (PESGM)*, 2026, pp. 1–5.
- [34] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1724–1734. [Online]. Available: <https://aclanthology.org/D14-1179/>
- [35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [36] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.
- [37] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "itransformer: Inverted transformers are effective for time series forecasting," *arXiv preprint arXiv:2310.06625*, 2023.
- [38] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations*, 2023. [Online]. Available: <https://arxiv.org/abs/2211.14730>
- [39] A. F. Ansari *et al.*, "Chronos-2: From univariate to universal forecasting," *arXiv:2510.15821*, 2025. [Online]. Available: <https://arxiv.org/abs/2510.15821>
- [40] A. Das, W. Kong, R. Sen, and Y. Zhou, "A decoder-only foundation model for time-series forecasting," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, 2024, pp. 10 148–10 167. [Online]. Available: <https://proceedings.mlr.press/v235/das24c.html>
- [41] Google Research, "TimesFM: Official implementation and TimesFM 2.5 release documentation," GitHub repository, accessed September 27, 2026. [Online]. Available: <https://github.com/google-research/timesfm>