DualEncDecoder: Federated Short-Term Load Forecasting under Heterogeneous Data[*]

Sasmita Harini S[1], **Naman Srivastava**[1], **Priyanka Nihalchandani**[1], **Varun Ojha**[2], **Pandarasamy Arjunan**[1]

[1]Indian Institute of Science (IISc) Bengaluru
[2]Newcastle University, UK
{sasmitah, snaman, priyankan, samy}@iisc.ac.in,
varun.ojha@newcastle.ac.uk

Abstract

Accurate short-term load forecasting (STLF) is a critical application for modern energy systems, enabling efficient scheduling, demand response, and grid reliability. Traditional centralized approaches raise privacy concerns and often degrade under heterogeneous building-level data. To address these challenges, this study explores federated learning (FL) with a novel *DualEncDecoder* architecture, a Difficulty-Aware Sampling (DAS) strategy, and an autoencoder-based clustering method for client grouping. Experimental results show that DualEncDecoder consistently outperforms baseline models, achieving the lowest SMAPE, while the clustering improves performance for simpler models such as LSTM and GRU. DualEncDecoder, also surpass centralized models under highly varying client data. Overall, the integration of advanced architectures, adaptive sampling, and representation-driven clustering establishes a robust framework for federated energy forecasting in privacy-sensitive, heterogeneous environments.

Introduction

The growing demand for electricity, driven by population growth, urbanization, and economic development, has elevated the accurate prediction of building energy consumption to the level of a critical application. Reliable energy forecasting directly impacts the stability of modern power systems, where supply-demand imbalances can lead not only to economic inefficiencies but also to risks of grid instability and power outages. In this context, energy forecasting is essential for maintaining resilience in critical infrastructures, supporting grid reliability, and ensuring sustainable energy distribution. Accurate forecasts enable effective resource utilization, informed operational planning, and proactive demand-side management (Mathumitha, Kumar, and Rajesh 2024).

Energy forecasting is typically categorized by prediction horizons, aligning with operational requirements of energy systems. *Short-term load forecasting* (STLF), which spans from one hour to one day, plays a particularly critical role. It supports daily scheduling, real-time demand response, and the integration of renewable energy, while also helping reduce costs and maintain occupant comfort in smart buildings (Abdulla, Almazrouei, and Alnuaimi 2024).

To achieve these goals, a variety of prediction models have been developed. Traditional approaches rely on statistical techniques, while recent advances leverage deep learning methods. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), are widely adopted due to their ability to capture temporal dependencies in energy consumption. More recently, foundation models have emerged as promising candidates for time series forecasting, owing to their robustness and generalizability across diverse domains (Saravanan, Kumar, and Lee 2024).

However, traditional centralized machine learning frameworks depend on aggregating data from multiple sources into a central server for global model training. While effective, this approach introduces significant privacy risks and often conflicts with data protection regulations. These limitations motivate the adoption of *Federated Learning* (FL), a decentralized paradigm that enables collaborative model training without exposing raw data. In FL, each client (e.g., a building or smart meter) trains locally on its private data and transmits only model updates to a central server, as illustrated in Figure 1. This architecture is particularly relevant for energy forecasting, where consumption patterns often contain sensitive information about occupant behavior.

Despite its promise, baseline FL algorithms such as FedAvg (McMahan et al. 2017) struggle in heterogeneous (non-IID) environments, which are common in real-world energy systems (Li et al. 2020b). Client drift,where local updates diverge from the global optimum-slows convergence and reduces prediction accuracy, undermining the reliability of forecasts in critical settings. To address this, methods like FedProx (Li et al. 2020a) and SCAFFOLD (Karimireddy et al. 2021) introduce mechanisms to mitigate divergence, while more recent work explores adaptive *client selection strategies* (Vardhan, Gupta, and Kumar 2025; Zhu, Zhang, and Bai 2024; Nishio and Yonetani 2019). For example, Active Federated Learning (AFL) (Goetz et al. 2019) prioritizes clients based on informativeness, and other studies (Jee Cho,

[*]Accepted at the 1st Workshop on Federated Learning for Critical Applications (FLCA), in conjunction with AAAI26

Wang, and Joshi 2022) suggest selecting clients with higher local loss, as their updates can disproportionately improve global accuracy.

In this setting, energy forecasting for smart buildings represents not just an optimization challenge but a critical application domain where data privacy, system robustness, and prediction accuracy intersect. Developing federated methods that can operate effectively under heterogeneity while safeguarding sensitive data is therefore central to ensuring reliable and secure energy management in modern infrastructure.

The main contributions of this work are as follows:

1. We propose a novel *difficulty-aware client sampling* (DAS) strategy for federated learning, which dynamically adapts the client selection process based on the learnability of each client's time series data, thereby improving convergence and fairness under non-IID settings.
2. We develop an *autoencoder-based clustering* framework to group clients with similar data representations, enabling a more balanced and effective aggregation process in federated environments.
3. We conduct a comprehensive empirical evaluation on a large-scale dataset comprising **1,410 real-world building energy meters** for short-term energy load forecasting, benchmarking DAS against the Power of Choice (POC) strategy under multiple deep learning architectures.
4. We demonstrate that our proposed *Dual-Encoder-Decoder* (DualEncDecoder) model consistently outperforms baseline approaches, leveraging covariate information and robust temporal modeling to achieve state-of-the-art forecasting performance in federated settings.

Background and Related Works

In federated learning (FL), the common assumption that all clients participate in every round rarely holds in practice due to device constraints, connectivity, or energy limitations. More realistic FL frameworks adopt partial participation, where only a subset of clients is selected per round (Fig. 1). Traditional approaches rely on random selection, implicitly assuming IID data across clients. However, real-world data are non-IID with varying dataset sizes, leading to degraded performance such as slower convergence, reduced accuracy, and fairness issues. Prior work has shown that intelligent client selection improves convergence and global performance (Lu et al. 2023; Goetz et al. 2019). For instance, Cho et al. (Jee Cho, Wang, and Joshi 2022) prioritized high-loss clients, while FedCor (Tang et al. 2022) introduced correlation-aware sampling via Gaussian Processes.

Load Forecasting Models

Load forecasting has progressed from classical statistical approaches to increasingly expressive deep learning models. Early methods such as *ARIMA* (Box and Jenkins 1976) offered interpretability but were limited in their ability to model nonlinear and long-range temporal dependencies. Recurrent neural architectures, including *RNNs* (Hewamalage, Bergmeir, and Bandara 2021), *LSTMs* (Alharthi and Hochreiter 2024), and *GRUs* (Choi and Kim 2020), addressed these limitations by explicitly modeling sequential dynamics. Convolutional neural networks and hybrid CNN–LSTM architectures further improved performance by capturing local temporal patterns and spatial correlations across features (Zhang, Zhu, and Bai 2022). Ensemble-based approaches, such as *Mixture of Experts*, have also been explored to enhance robustness and adaptability in heterogeneous forecasting scenarios (Reisser, Louizos, and Gavves 2021).

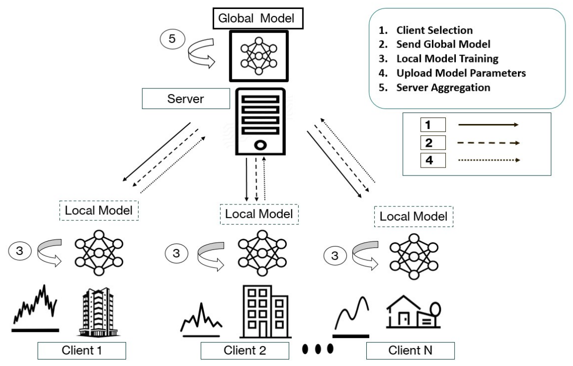

Figure 1: Federated Learning Architecture

Beyond purely historical load signals, recent studies have emphasized the importance of incorporating exogenous covariates such as temperature, weather conditions, and building usage characteristics to improve forecasting accuracy and generalization. These auxiliary inputs provide contextual information that helps disambiguate load variations driven by external factors rather than intrinsic temporal dynamics alone. More recently, transfer learning and pre-trained foundation models, including large language model (LLM)-based and time-series-specific architectures, have been applied to load forecasting tasks, demonstrating improved performance across diverse datasets and deployment settings (Kumar, Singh, and Gupta 2025; Liu, Zhang, and Chen 2024; Saravanan, Kumar, and Lee 2024). Together, these developments highlight a shift toward context-aware and transferable forecasting models capable of leveraging both historical load patterns and rich covariate information.

Federated Learning Foundations

FL (Kairouz et al. 2021) enables decentralized training by aggregating client updates while keeping data local, ensuring privacy and reducing communication. FedAvg (McMahan et al. 2017) remains the foundational algorithm, though it struggles in heterogeneous data settings (Li et al. 2022). Extensions include FedProx (Li et al. 2020a), which stabilizes updates with a proximal term, SCAFFOLD (Karimireddy et al. 2021), which uses control variates to reduce client drift, and FedNova (Wang et al. 2020), which normalizes updates for varying local steps. FL has also seen growing adoption in energy applications (Zhang, Li, and Chen 2023).

Client Selection Strategies in FL

Client selection plays a key role in improving convergence and fairness in FL. Early methods such as FedCS (Tang et al. 2022) focused on availability and computational capacity but overlooked data heterogeneity. More recent strategies prioritize high-loss clients (Zhu, Zhang, and Bai 2024; Vardhan, Gupta, and Kumar 2025), though this may amplify noise or outliers. Adaptive approaches address this by jointly considering data diversity and system constraints (e.g., latency, communication cost). Vardhan et al. (Vardhan, Gupta, and Kumar 2025), for example, frame client selection as an optimization problem balancing round latency with pairwise data dissimilarity, achieving faster convergence and better generalization in realistic settings.

Clustering strategies in FL

In federated learning (FL) with heterogeneous clients arising from non-IID local data, a widely adopted strategy is to partition clients into clusters with similar data distributions or tasks, followed by training cluster-specific models rather than a single global model. For example, (Ghosh et al. 2020) introduce the Iterative Federated Clustering Algorithm (IFCA), which alternates between cluster assignment and optimization of cluster-specific models, and establish improved convergence guarantees under data heterogeneity. Likewise, (Long et al. 2023) present a multi-center FL framework that jointly learns multiple global models and assigns clients through an Expectation–Maximization procedure, demonstrating gains in both personalization and predictive accuracy.

Within energy and smart-metering applications, clustering has been leveraged for load-forecasting tasks. Gholizadeh and Musilek (Gholizadeh and Musilek 2021) propose a federated learning framework that clusters clients based on hyper-parameter dynamics (e.g., learning rate, batch size, and epoch behavior) rather than raw consumption profiles, thereby mitigating the impact of heterogeneous data distributions on convergence. Their results show accelerated convergence and enhanced model accuracy.

In addition, embedding time-series data into latent representations has been shown to improve clustering quality. For instance, (Cortés et al. 2024) combine autoencoders with clustering to capture temporal and structural dependencies in financial time series, yielding more coherent groupings. Motivated by these findings, our proposed framework employs autoencoder-derived embeddings, together with synthetic augmentation, to cluster clients in a privacy-preserving FL setting. This design aims to reduce client drift induced by distributional heterogeneity and to improve the efficiency of federated training for energy consumption forecasting.

Federated Learning for Time Series Forecasting

Applying FL to time-series forecasting introduces additional complexities due to temporal dependencies, variable data quality, and inconsistent patterns across clients. Standard FL models often underperform in this context because of the high degree of non-stationarity and client-specific seasonality. To address these challenges, several studies have extended FL to handle such settings by incorporating time-aware architectures such as LSTM and hybrid CNN-LSTM models (Zhang, Zhu, and Bai 2022). Moreover, the use of personalized models (Wang and Kantarci 2024) and dynamic aggregation strategies, and client clustering (Li, Zhang, and Smith 2025) has shown promise in adapting client-level variability.

Despite these advancements, the interplay between temporal modeling, client selection, and heterogeneous data distributions remains insufficiently explored. This motivates the need for adaptive strategies that not only respect the sequential nature of time series data but also account for client reliability and informativeness during training.

Challenges in Federated Learning

Federated learning (FL) introduces several challenges that hinder robust deployment in real-world settings. A primary difficulty arises from the presence of non-IID data across clients, where heterogeneous local data distributions can bias model updates and adversely affect convergence and generalization (Li et al. 2022). In addition, system heterogeneity, including variations in computational resources, communication latency, and device availability, further complicates synchronized training and scalable participation. As a result, the selection of participating clients in each communication round plays a critical role in balancing statistical representativeness, training efficiency, and system scalability. Despite extensive research, learning a global model that performs consistently well across diverse client distributions remains an open problem.

To address these challenges, prior work has explored adaptive client selection strategies that account for data heterogeneity and temporal dynamics. In particular, difficulty-aware and curriculum-inspired approaches aim to order client participation based on learnability, allowing models to stabilize on simpler or more informative clients before incorporating more complex or noisy ones. Related efforts have also considered concept drift through probabilistic client sampling mechanisms. For instance, (Zhu, Zhang, and Bai 2024) dynamically adjusts client selection probabilities based on observed local training loss, enabling responsiveness to evolving data distributions over time. These approaches highlight the utility of client-side loss as an informative and low-overhead signal for guiding client selection in federated optimization, motivating its use in federated time series forecasting settings.

Dataset and Preprocessing

This study employs the ASHRAE Great Energy Predictor III dataset (Miller et al. 2020), which contains hourly energy consumption records from over 1,400 commercial buildings across multiple sites in the United States. Each record includes a timestamp, building identifier, meter type and energy usage. We focus solely on electricity consumption, yielding 1,410 buildings after filtering. In line with established benchmarking protocols, we discard electricity readings recorded before May 21, 2016, for certain buildings due to known anomalies. For `site_id = 0`, meter readings originally reported in kBTU are converted to kilowatt-hours using a standard factor of 0.2931.

To ensure consistency across buildings, identifiers are reindexed to a continuous range from 0 to 1,409. Any missing values introduced during this alignment are imputed using linear interpolation with both forward and backward filling.

Methodology

This study proposes a novel Short-Term Load Forecasting (STLF) system for building energy time series data, leveraging federated learning (FL) with three key innovations: (1) Difficulty-Aware Sampling for client selection, (2) Client Clustering using Autoencoder-based embeddings, and (3) a novel `DualEncDecoder` model architecture for forecasting. The methodology integrates synthetic time series generation, autoencoder-based feature extraction, client clustering, and federated training to achieve robust and adaptive load forecasting.

Overview of the Pipeline

The Short-Term Load Forecasting (STLF) pipeline integrates synthetic data generation, client clustering, and federated learning to achieve robust and privacy-preserving energy forecasting. The pipeline comprises the following stages:

1. **Clustering**: Synthetic time series are generated to mimic real energy load patterns while preserving privacy. These series are decomposed using Seasonal-Trend Decomposition (STL) into trend, seasonal, residual, and covariate components. Autoencoders are trained on these components to extract latent representations, which are used to cluster clients based on load pattern similarity, as detailed in Algorithm 1.
2. **Federated Learning with Client Sampling**: After clustering, federated learning begins by selecting a subset of clients from each cluster using the Difficulty-Aware Sampling (DAS) strategy. Selected clients train local models on their private data and send model updates to a central server for aggregation using the FedAvg algorithm.
3. **Forecasting and Evaluation**: After training, the global model generates 24-step-ahead forecasts. Performance is evaluated using the Symmetric Mean Absolute Percentage Error (SMAPE) and Normalized Root Mean Square Error (NRMSE).

Client Sampling: Difficulty-Aware Sampling

In traditional federated learning, client selection is often random or uniform, which can lead to inefficient training when clients exhibit heterogeneous data distributions and learning dynamics. To address this issue, we propose a *Difficulty-Aware Sampling (DAS)* strategy that adaptively prioritizes clients based on their observed training difficulty.

Our approach is inspired by the general notion of training difficulty introduced in (Wang and Li 2023), but does not directly adopt their formulation. Instead, we redefine difficulty at the client level using local loss dynamics and integrate it into a probabilistic sampling mechanism tailored to partial client participation in federated learning.

Let N denote the total number of clients, and let $\mathcal{S}_t \subset \{1, \dots, N\}$ denote the set of m clients selected for local training in communication round t.

Loss definition. For a client $i \in \mathcal{S}_t$, let $\ell_t^{(i)}$ denote the local training loss observed after completing its local update in round t. If client i was not selected in the immediately preceding round $t-1$, we define

$$\ell_{t-1}^{(i)} \triangleq \ell_{\tau_i(t)}^{(i)},$$

where

$$\tau_i(t) = \max\{\tau < t \mid i \in \mathcal{S}_\tau\}$$

denotes the most recent round prior to t in which client i participated. Thus, $\ell_{t-1}^{(i)}$ always refers to the *last observed loss* of client i. Importantly, $\ell_{t-1}^{(i)}$ does *not* correspond to the loss at the immediately preceding global round $t-1$, but rather to the most recent loss observed for client i when it last participated in training. For clients that have never participated, $\ell_0^{(i)}$ is initialized to a constant.

Loss dynamics. The change in loss for client i at round t is defined as

$$\Delta\ell_t^{(i)} = \ell_t^{(i)} - \ell_{t-1}^{(i)}.$$

For numerical stability, we further compute the logarithmic loss ratio

$$r_t^{(i)} = \log\left(\frac{\ell_t^{(i)} + \varepsilon}{\ell_{t-1}^{(i)} + \varepsilon}\right),$$

where $\varepsilon > 0$ is a small constant preventing division by zero.

Learnability and unlearnability. Based on the sign of the loss change, we define two complementary scores:

$$\text{learn}_t^{(i)} = \begin{cases} -\Delta\ell_t^{(i)} \cdot r_t^{(i)}, & \text{if } \Delta\ell_t^{(i)} < 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$\text{unlearn}_t^{(i)} = \begin{cases} \Delta\ell_t^{(i)} \cdot r_t^{(i)}, & \text{if } \Delta\ell_t^{(i)} \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

The learnability score captures effective training progress, while the unlearnability score reflects stagnation or degradation in local optimization.

Temporal smoothing. To reduce noise caused by round-to-round fluctuations, both scores are smoothed using exponential moving averages (EMAs):

$$L_t^{(i)} = \alpha \cdot \text{learn}_t^{(i)} + (1-\alpha)L_{t-1}^{(i)},$$

$$U_t^{(i)} = \alpha \cdot \text{unlearn}_t^{(i)} + (1-\alpha)U_{t-1}^{(i)},$$

where $\alpha \in (0,1)$ controls the smoothing factor.

Difficulty estimation. The instantaneous difficulty of client i is defined as

$$D_t^{(i)} = \frac{U_t^{(i)} + \varepsilon}{L_t^{(i)} + \varepsilon}.$$

To obtain a stable difficulty estimate over time, an additional EMA is applied:

$$\hat{D}_t^{(i)} = \alpha \cdot D_t^{(i)} + (1-\alpha)\hat{D}_{t-1}^{(i)}.$$

Sampling probabilities. Clients are sampled according to the inverse of their estimated difficulty:

$$\tilde{p}_i = \frac{1}{\hat{D}_t^{(i)} + \varepsilon}, \qquad p_i = \frac{\tilde{p}_i}{\sum_{j=1}^{N} \tilde{p}_j}.$$

To prevent starvation of difficult clients, a minimum probability threshold δ is enforced:

$$p_i' = \max(p_i, \delta), \qquad \hat{p}_i = \frac{p_i'}{\sum_{j=1}^{N} p_j'}.$$

The final probabilities $\hat{p}_i$ are used to sample the client set $\mathcal{S}_{t+1}$ for the next communication round.

This difficulty-aware sampling strategy prioritizes clients that are easier to optimize while ensuring that all clients remain eligible for participation, thereby balancing training efficiency and fairness.

Client Clustering Using Autoencoders

To account for heterogeneity in client data (e.g., different building types or usage patterns) while preserving the privacy of sensitive load data, we propose a novel clustering approach using autoencoder-based embeddings derived from synthetic data, as detailed in Algorithm 1. This approach ensures that actual client load data remains local and private, while still enabling effective clustering based on representative embeddings.

Synthetic Data Generation To protect the privacy of building energy load data, synthetic time series are generated to mimic the statistical properties of real data without exposing sensitive information. Each series is then decomposed using STL (Seasonal-Trend Decomposition) using LOESS - Locally Estimated Scatterplot Smoothing (Cleveland et al. 1990)).

Autoencoder Training Four autoencoders (Trend AE, Seasonal AE, Residual AE, Covariate AE) are trained on synthetic data (for time series components) and real covariate data, with early stopping based on validation loss. Covariates (air temperature and primary use) are OneHot encoded and normalized before training. All use ReLU activations and fully connected layers.

Client Embedding and Clustering This enables grouping of clients by load pattern similarity, improving downstream federated learning strategies.

Algorithm 1: Privacy-Preserving Client Clustering via Synthetic Data and Autoencoders

Input: Number of synthetic series $n_{\text{synthetic}}$, series length $L = 168$, period $P = 24$, client data
Parameter: Noise levels $\sigma_t, \sigma_s, \sigma_r$, autoencoder architecture, number of clusters K
Output: Clustered client groups $\{C_1, C_2, \dots, C_K\}$

```
1:  // Step 1: Synthetic Time Series Generation
2:  for $i = 1$ to $n_{\text{synthetic}}$ do
3:      $t_i \leftarrow 0.05 \cdot \text{arange}(L) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_t^2)$
4:      $s_i \leftarrow 0.5 \cdot \sin(2\pi k/P) + \epsilon_s, \quad \epsilon_s \sim \mathcal{N}(0, \sigma_s^2)$
5:      $r_i \leftarrow \epsilon_r, \quad \epsilon_r \sim \mathcal{N}(0, \sigma_r^2)$
6:      $x_i \leftarrow t_i + s_i + r_i$
7:  end for
8:  // Step 2: STL Decomposition
9:  for each $x_i$ do
10:     $(\text{trend}_i, \text{season}_i, \text{resid}_i) \leftarrow \text{STL}(x_i)$
11:     $\text{resid}_i \leftarrow x_i - \text{trend}_i - \text{season}_i$
12: end for
13: // Step 3: Autoencoder Training
14: for each component $\in \{\text{trend}, \text{season}, \text{resid}, \text{cov}\}$ do
15:     Train AE with encoder $f_\theta$, decoder $g_\phi$ to minimize:
            $\mathcal{L}_{AE} = \frac{1}{N}\sum_{i=1}^{N} \|x_i - g_\phi(f_\theta(x_i))\|_2^2$
16: end for
17: // Step 4: Client Embedding (Local)
18: for each client $c$ do
19:     Apply STL on $x_c$: $(\text{trend}_c, \text{season}_c, \text{resid}_c)$
20:     Encode components using trained AEs:
            $z_c = [f_t(\text{trend}_c), f_s(\text{season}_c), f_r(\text{resid}_c), f_{\text{cov}}(\text{cov}_c)]$
21:     Send normalized $z_c$ to server
22: end for
23: // Step 5: Server-Side Clustering
24: Apply KMeans on $\{z_c\}$ to form $K$ clusters:
            $\min_{\{C_k\}} \sum_{k=1}^{K} \sum_{z_c \in C_k} \|z_c - \mu_k\|_2^2$
25: return $\{C_1, C_2, \dots, C_K\}$
```

Dual Encoder-Decoder (DualEncDecoder) Model

We introduce a novel forecasting model, `DualEncDecoder`, designed to handle time series data with covariates and categorical features. The model combines 2 convolutional neural networks (CNNs), gated recurrent unit (GRU), and a fully connected decoder, with an embedding layer for categorical variables as shown in Figure 2. Time-series load data includes meter readings and STL-decomposed components (trend, seasonal, residual). The model produces a 24-step-ahead forecast for each input sequence, suitable for short-term load forecasting.

Experiments

We conducted extensive experiments to evaluate the proposed Short-Term Load Forecasting (STLF) system using the ASHRAE Great Energy Predictor III dataset. The experiments assess the performance of the `DualEncDecoder` model with Difficulty-Aware Sampling (DAS) against baseline models and sampling strategies, focusing on Symmetric

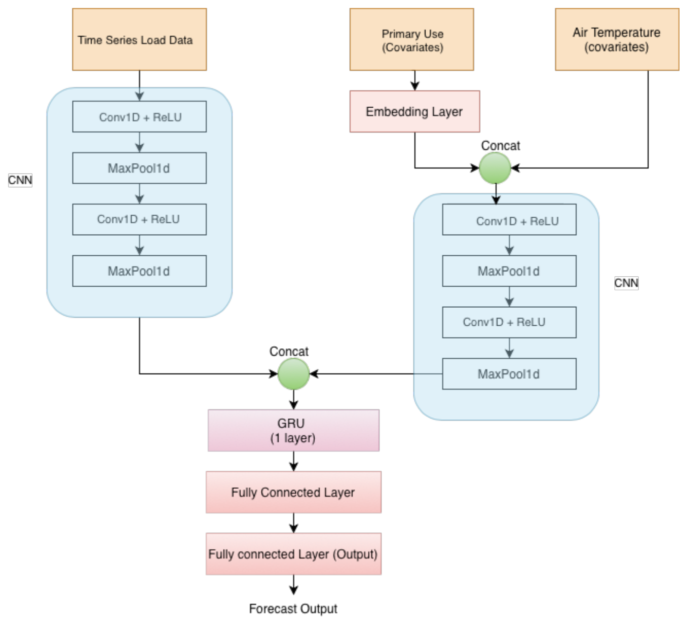

Figure 2: Proposed Dual Encoder Decoder Model Architecture

Mean Absolute Percentage Error (SMAPE) and Normalized Root Mean Square Error (NRMSE). The best results in each table are **bolded** for clarity.

Experimental Setup

The experimental evaluation is conducted on a dataset comprising 1,410 buildings, each treated as an individual client within the federated learning framework. To emulate realistic participation constraints, only 15% of clients are sampled in each communication round. Training proceeds for 40 rounds, with each selected client performing five local epochs of optimization using a learning rate of 0.001. For comparisons, centralised models are trained on all client data. To ensure fair comparison, the centralized baseline uses 30 epochs, approximating the effective local training iterations $(0.15 \times 40 \times 5)$ in FL. We compare the proposed Difficulty-Aware Sampling (DAS) strategy against the established Power of Choice (POC) method (Jee Cho, Wang, and Joshi 2022), both implemented under the standard FedAvg aggregation algorithm. A diverse set of deep learning models is evaluated, ranging from simpler feedforward architectures to recurrent and hybrid designs, including a shallow fully connected feedforward neural network (FCNN), LSTM, GRU, CNN-GRU (without covariates), a Mixture-of-Experts LSTM (MoE-LSTM) model, and the proposed `DualEncDecoder`. Model configuration details are provided in the Appendix. To account for the heterogeneity inherent in non-IID client data, we report results using the median error across clients, which better reflects robustness and fairness compared to mean-based evaluation.

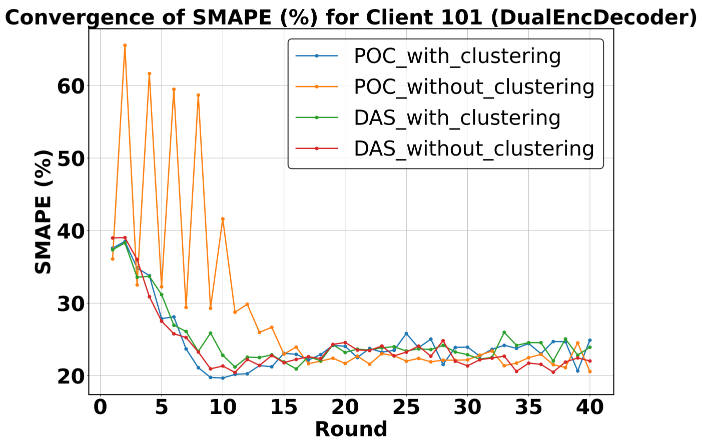

Figure 3: Convergence plot for proposed model and Sampling Strategy for select client

Results

Tables 1 and 2 report the forecasting performance of all models under different client sampling strategies, evaluated using SMAPE (%) and NRMSE (%), respectively (metric definitions are provided in the appendix). Across both error metrics and all federated configurations, the proposed `DualEncDecoder` consistently outperforms all baseline models, demonstrating strong robustness to client heterogeneity and sampling variability.

Under SMAPE, `DualEncDecoder` achieves its best federated performance with DAS (No Clustering) at 22.9394%, closely followed by POC (No Clustering) at 22.7855%. A similar trend is observed under NRMSE, where `DualEncDecoder` attains the lowest error of 26.62% under both DAS (Clustering) and POC (Clustering), indicating stable performance across sampling strategies. In contrast, recurrent and convolutional baselines exhibit substantially higher errors under both metrics. Notably, `DualEncDecoder` also significantly outperforms all centralized baselines, highlighting its strong generalization capability.

For covariate-aware architectures such as CNN-GRU, clustering yields marginal or inconsistent improvements across both metrics, suggesting limited benefit when covariate information is already explicitly modeled. In contrast, simpler or less expressive architectures benefit more consistently from clustering. Models such as MOE-LSTM, LSTM, and GRU generally achieve lower or more stable errors under clustering-based DAS or POC, particularly under NRMSE, indicating reduced sensitivity to client heterogeneity.

Architectures with dual-branch designs exhibit mixed behavior. While `Dual CNN ANN` benefits from POC-based sampling in both metrics, its performance remains inferior to `DualEncDecoder`, underscoring the importance of encoder–decoder structure for capturing complex temporal dependencies in federated environments. Figure 3 depict the convergence across aggregation rounds for a select client.

Figures 4 and 5 illustrate the mean annual and aver-

Table 1: Comparison of Sampling Strategies Across Models using SMAPE (%)

Model	**DAS**		**POC**		**Centralised**
	Clustering	**No Clustering**	**Clustering**	**No Clustering**	**Model**
AutoARIMA	NA	NA	NA	NA	33.55
FCNN	28.44	28.54	28.43	28.42	27.68
CNN	28.44	28.43	28.59	28.75	29.67
GRU	30.79	30.75	30.69	31.07	26.57
LSTM	31.80	33.28	31.70	32.30	26.43
CNN-GRU	30.94	31.10	30.87	31.47	27.18
MOE-LSTM	32.24	32.50	32.92	32.64	26.82
Dual FCNN	24.12	23.79	24.57	26.82	21.27
Dual CNN ANN	29.92	27.33	24.65	24.86	14.96
DualEncDecoder (Proposed)	**23.33**	**22.94**	**23.81**	**22.79**	**8.44**

Table 2: Comparison of Sampling Strategies Across Models using NRMSE (%)

Model	**DAS**		**POC**		**Centralised**
	Clustering	**No Clustering**	**Clustering**	**No Clustering**	**Model**
AutoARIMA	NA	NA	NA	NA	41.34
FCNN	32.59	32.50	32.43	32.35	31.69
CNN	32.45	32.46	32.45	32.52	33.15
GRU	34.05	35.07	34.05	34.21	31.19
LSTM	34.77	35.64	34.64	34.97	31.24
CNN-GRU	33.92	34.01	33.92	34.32	31.48
MOE-LSTM	34.94	35.10	35.31	35.56	31.39
Dual FCNN	29.01	28.61	29.31	33.04	25.31
Dual CNN ANN	35.76	31.93	28.86	28.82	17.29
DualEncDecoder (Proposed)	**26.62**	**27.65**	**26.62**	**26.65**	**10.04**

age daily load patterns for representative client clusters. The plots reveal pronounced heterogeneity across clusters, with distinct seasonal trends over the year and characteristic intra-day consumption profiles. Some clusters exhibit strong seasonal variability, while others maintain relatively stable demand, reflecting diverse consumption behaviors across clients.

Figure 6 presents a t-SNE projection of client embeddings learned during federated training. Each point corresponds to a client, colored by cluster assignment. The visualization shows the groups in the embedding space, indicating that clients with similar temporal load characteristics are mapped to nearby representations. Partial overlap between some clusters suggests gradual transitions in client behavior rather than rigid boundaries, further highlighting the structured yet heterogeneous nature of the client population.

At a finer temporal scale, Figure 5 demonstrates substantial variation in diurnal load shapes across clusters, including differences in peak timing, magnitude, and dispersion throughout the day. Together, these observations confirm that the clustering captures meaningful structural differences in both long-term and short-term load dynamics, providing a strong empirical basis for clustering-aware analysis in federated energy forecasting.

Discussion

The experimental results demonstrate that the proposed `DualEncDecoder` consistently outperforms all baseline models across both SMAPE and NRMSE, highlighting its ability to model complex temporal patterns under non-IID federated settings. Its best performance is achieved with DAS without clustering, indicating that for expressive encoder–decoder architectures, explicit client clustering offers limited additional benefit. In contrast, clustering improves or stabilizes performance for models such as MOE-LSTM and GRU, which benefit from reduced inter-client variability.

The cluster-level visualizations in Figures 4 and 5 provide important insight into these trends. The figures reveal substantial heterogeneity across client clusters, with distinct seasonal behaviors over the year and markedly different diurnal load patterns. This structural diversity explains why clustering can be beneficial for less expressive models, while more powerful architectures such as `DualEncDecoder` can implicitly adapt to heterogeneous client distributions without explicit grouping.

Across sampling strategies, DAS performs on par with or better than PoC while being significantly more efficient. As shown in Appendix , PoC introduces an additional per-round computation cost of $O(d \cdot F)$ due to extra inference-only evaluations of the global model on candidate clients. In contrast, DAS incurs only $O(N)$ scalar operations per round and

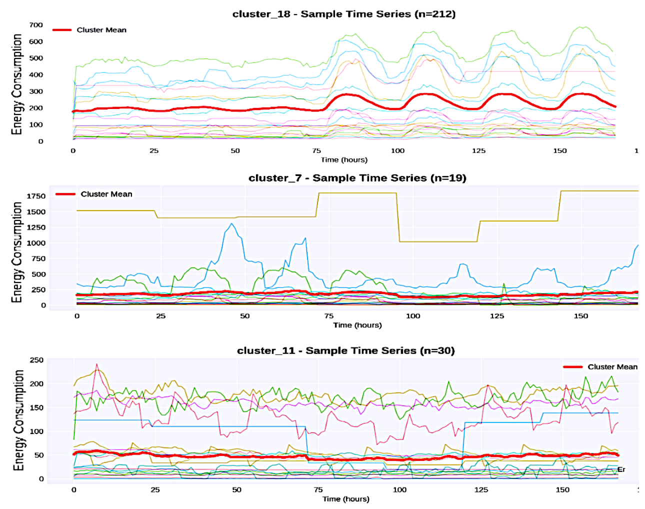

Figure 4: Visualization of mean annual load profiles for select client clusters

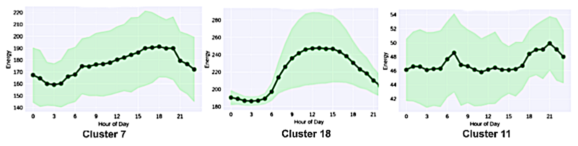

Figure 5: Visualization of average daily load profiles for select client clusters

requires no additional model forward passes, as it reuses loss values obtained during standard local training. The comparable or superior accuracy of DAS observed in our experiments therefore comes without the computational overhead inherent to PoC.

Taken together, these results suggest that difficulty-aware sampling provides a more scalable and practical alternative to power-of-choice sampling for large, heterogeneous client populations. When combined with an expressive model such as `DualEncDecoder`, DAS offers a robust federated learning framework that balances forecasting accuracy, computational efficiency, and adaptability to diverse load patterns.

Conclusion

This work presented a comprehensive study on federated time series forecasting under heterogeneous client distributions, introducing contributions at the levels of model design, sampling strategy, and clustering. First, we proposed the `DualEncDecoder` architecture, which demonstrated superior performance compared to baseline models, effectively capturing complex temporal dynamics and covariate interactions in non-IID data. Second, we introduced the Difficulty-Aware Sampling (DAS) strategy, which proved competitive with and often superior to the Power of Choice (POC) approach, offering improvements in forecasting accuracy by prioritizing clients with stable learning progress. Finally, we examined the role of clustering in federated learning, showing that while clustering enhances performance for models such as MOE-LSTM and GRU, it does not necessarily benefit more expressive architectures like `DualEncDecoder`.

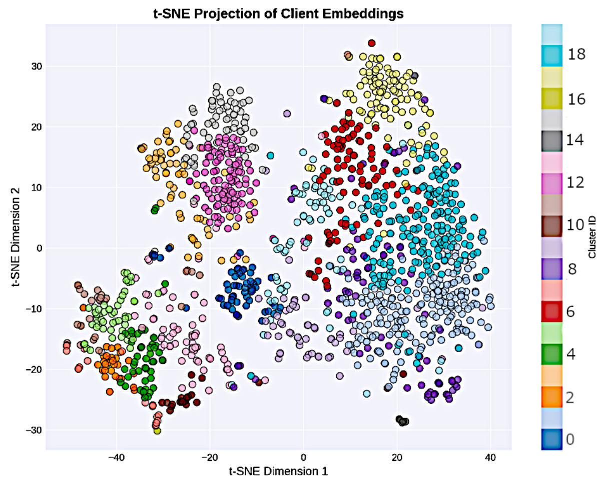

Figure 6: t-SNE projection of learned client embeddings

Taken together, these contributions highlight that the combination of advanced model design and informed client sampling can significantly improve federated forecasting. Moreover, the findings suggest that clustering remains a promising direction for enhancing convergence and fairness in certain scenarios, though its effectiveness depends on the underlying model capacity. Future work will explore adaptive strategies that integrate clustering with model-informed sampling to further advance federated learning for real-world time series applications.

Acknowledgments

This work was supported by a research grant from the AI & Robotics Technology Park (ARTPARK) at the Indian Institute of Science. The authors gratefully acknowledge the dataset providers for their efforts in making the data publicly available and maintaining high-quality resources for this study

References

Abdulla, K.; Almazrouei, S.; and Alnuaimi, A. 2024. A novel approach to short-term load forecasting in smart buildings using federated learning. *Energy and Buildings*, 301: 101342.

Alharthi, M.; and Hochreiter, S. 2024. xLSTMTime: Long-term time series forecasting with xLSTM. *arXiv preprint arXiv:2405.06846*.

Box, G. E. P.; and Jenkins, G. M. 1976. Short-term load forecasting using ARIMA model. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(3): 337–352.

Choi, B. J.; and Kim, J. Y. 2020. Short-term load forecasting using GRU networks. 1–6.

Cleveland, R. B.; Cleveland, W. S.; McRae, J. E.; Terpenning, I.; et al. 1990. STL: A seasonal-trend decomposition. *J. off. Stat*, 6(1): 3–73.

Cortés, D. G.; Onieva, E.; López, I. P.; Trinchera, L.; and Wu, J. 2024. Autoencoder-Enhanced Clustering: A Dimensionality Reduction Approach to Financial Time Series. *IEEE Access*, 12: 16999–17009.

Gholizadeh, N.; and Musilek, P. 2021. Federated learning with hyperparameter-based clustering for electrical load forecasting. *Internet of Things*, 17: 100470.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An efficient framework for clustered federated learning.

Goetz, J.; Malik, K.; Bui, D.; Moon, S.; Liu, H.; and Kumar, A. 2019. Active federated learning. *arXiv preprint arXiv:1909.12641*.

Hewamalage, H.; Bergmeir, C.; and Bandara, K. 2021. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1): 388–427.

Jee Cho, Y.; Wang, J.; and Joshi, G. 2022. Power of choice in federated learning. *Proceedings of Machine Learning Research*, 151: 5869–5888.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S. J.; Stich, S. U.; and Suresh, A. T. 2021. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. arXiv:1910.06378.

Kumar, R.; Singh, P.; and Gupta, A. 2025. MixForecast: A federated learning approach for enhanced time series prediction. *IEEE Transactions on Artificial Intelligence*, 6(2): 345–357.

Li, T.; Sahu, A. K.; Talwalkar, A.; Smith, V.; Allouah, A.; Konečný, J.; Zhang, W.; Kumar, J.; Andrew, G.; Wang, P.; et al. 2020a. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2: 429–450.

Li, T.; Zhang, W.; and Smith, V. 2025. Federated learning with client clustering for personalized models. *IEEE Transactions on Artificial Intelligence*, 6(3): 789–802.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2020b. On the Convergence of FedAvg on Non-IID Data. arXiv:1907.02189.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2022. On the convergence of FedAvg on non-IID data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11): 8313–8326.

Liu, Y.; Zhang, W.; and Chen, M. 2024. TimeFFM: LLM-empowered federated foundation model for time series forecasting. *arXiv preprint arXiv:2403.12345*.

Long, G.; Tan, Y.; Jiang, J.; Zhang, C.; Liu, M.; and Zhou, X. 2023. Multi-center federated learning: Clients clustering for better personalization. *World Wide Web*, 26(4): 2163–2184.

Lu, Y.; Huang, X.; Xu, W.; and Jiao, Y. 2023. Client selection for federated learning in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 24(10): 10823–10834.

Mathumitha, R.; Kumar, S.; and Rajesh, T. 2024. Energy forecasting for smart buildings: A review of methods and applications. *Renewable and Sustainable Energy Reviews*, 189: 113892.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Miller, C.; Kathirgamanathan, A.; Picchetti, B.; Arjunan, P.; Park, J. Y.; Nagy, Z.; Raftery, P.; Hobson, B. W.; Shi, Z.; and Meggers, F. 2020. The building data genome project 2, energy meter data from the ASHRAE great energy predictor III competition. *Scientific data*, 7(1): 368.

Nishio, T.; and Yonetani, R. 2019. Client selection for federated learning with heterogeneous resources in mobile edge. *IEEE International Conference on Communications (ICC)*, 1–7.

Reisser, M.; Louizos, C.; and Gavves, E. 2021. Federated mixture of experts for time series forecasting. *arXiv preprint arXiv:2106.10345*.

Saravanan, V.; Kumar, A.; and Lee, J. 2024. Analyzing the efficacy of foundation models in time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6): 7890–7902.

Tang, T.; Li, S.; Zhang, W.; Zhu, S.; and Bai, X. 2022. FedCor: Correlation-based active client selection strategy for heterogeneous federated learning. *arXiv preprint arXiv:2112.14412*.

Vardhan, H.; Gupta, S.; and Kumar, R. 2025. Client selection in federated learning: Convergence analysis and optimization. *IEEE Transactions on Mobile Computing*, 24(1): 456–468.

Wang, H.; and Li, X. 2023. Dhc: Dual-debiased heterogeneous co-training framework for class-imbalanced semi-supervised medical image segmentation. *International conference on medical image computing and computer-assisted intervention*, 582–591.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33: 7611–7623.

Wang, N.; and Kantarci, B. 2024. Personalized federated learning for time series forecasting. *Future Generation Computer Systems*, 152: 114762.

Zhang, G.; Zhu, S.; and Bai, X. 2022. Federated learning-based multi-energy load forecasting method using CNN-attention-LSTM model. *Sustainability*, 14(19): 12843.

Zhang, W.; Li, J.; and Chen, M. 2023. Federated learning for energy management in smart grids. *IEEE Transactions on Smart Grid*, 14(5): 3762–3774.

Zhu, S.; Zhang, W.; and Bai, X. 2024. *Client Selection for Federated Learning: A Dynamic Systems Perspective*. Springer.

Appendix

This appendix provides detailed descriptions of the model architectures, additional experimental results, and supplementary materials to support the main text. All models are designed for Short-Term Load Forecasting (STLF) with an input sequence length of 168 hours and a forecast horizon of 24 hours, unless otherwise specified.

Model Architectures

Below, we detail the architectures of the models evaluated in the experiments: `DualEncDecoder`, `CNN-GRU`, `MOE-LSTM`, `LSTM`, `GRU`, `FCNN`, `AE`, `DualGRUGRU`, `DualGRUFCNN`, `CNN`, `DualFCNN`, `DualCNNANN`, `AutoARIMA`. Each model is described with its input/output shapes, layer configurations, and key hyperparameters.

DualEncDecoder The `DualEncDecoder` model integrates time-series and covariate data with positional encoding for robust forecasting. Its architecture is as follows:

- **Primary Use Embedding**: Maps 16 primary use categories to a 16-dimensional vector using an embedding layer (`[batch, 16]`).
- **Time-Series CNN**: Two 1D convolutional layers process the time-series input (`[batch, 168, 4]`):
  - Layer 1: 4 $\rightarrow$ 64 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Layer 2: 64 $\rightarrow$ 128 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Output: `[batch, 84, 128]`.
- **Covariate CNN**: Two 1D convolutional layers process the concatenated covariate and embedding input (`[batch, 168, 17]`):
  - Layer 1: 17 $\rightarrow$ 32 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Layer 2: 32 $\rightarrow$ 64 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Output: `[batch, 84, 64]`.
- **GRU**: A single-layer GRU (input size 192, hidden size 128) processes the concatenated CNN outputs (`[batch, 84, 192]`).
- **Output Layers**: Two linear layers (128 $\rightarrow$ 256, 256 $\rightarrow$ 24) with ReLU activation produce the 24-step forecast (`[batch, 24]`).

CNN-GRU The `CNN-GRU` model processes only time-series features (meter readings and STL components; 4 features) without covariates:

- **CNN**: A 1D convolutional layer (input channels 4 $\rightarrow$ 32, kernel size 3, padding 1, ReLU, max-pooling with kernel size 2) processes the input (`[batch, 168, 4]`). Output: `[batch, 84, 32]`.
- **GRU**: A single-layer GRU (input size 32, hidden size 64) processes the CNN output.
- **Output Layer**: A linear layer (64 $\rightarrow$ 24) maps the last GRU hidden state to the 24-step forecast (`[batch, 24]`).

MOE-LSTM The Mixture of Experts LSTM (`MOE-LSTM`) combines LSTM layers with a mixture of experts framework:

- **Lower LSTM**: A single-layer LSTM (input size 4, hidden size 64) processes the time-series input (`[batch, 168, 4]`).
- **Experts**: Six linear layers (64 $\rightarrow$ 32, ReLU) act as experts.
- **Gate**: A linear layer (64 $\rightarrow$ 6, softmax) generates weights for combining expert outputs.
- **Upper LSTM**: A single-layer LSTM (input size 32, hidden size 64) processes the weighted expert outputs.
- **Output Layers**: A linear layer (64 $\rightarrow$ 32, ReLU) followed by a linear layer (32 $\rightarrow$ 24) produces the forecast (`[batch, 24]`).

LSTM The `LSTM` model is a standard recurrent architecture:

- **LSTM**: A 2-layer LSTM (input size 4, hidden size 82, dropout 0.2) processes the time-series input (`[batch, 168, 4]`).
- **Output Layer**: A linear layer (82 $\rightarrow$ 24) maps the last hidden state to the 24-step forecast (`[batch, 24]`).

GRU The `GRU` model uses Gated Recurrent Units:

- **GRU**: A 2-layer GRU (input size 4, hidden size 82, dropout 0.2) processes the time-series input (`[batch, 168, 4]`).
- **Output Layer**: A linear layer (82 $\rightarrow$ 24) maps the last hidden state to the 24-step forecast (`[batch, 24]`).

FCNN The `FCNN` is a feedforward neural network:

- **Flatten**: Flattens the input (`[batch, 24, 4]`) to a 96-dimensional vector.
- **Hidden Layers**: Two linear layers (96 $\rightarrow$ 64, 64 $\rightarrow$ 64) with ReLU activations.
- **Output Layer**: A linear layer (64 $\rightarrow$ 8) produces an 8-step forecast (`[batch, 8]`).

Autoencoder (AE) The `Autoencoder` generates embeddings for client clustering:

- **Encoder**: Two linear layers (input dim $\rightarrow$ mid dim, mid dim $\rightarrow$ latent dim) with ReLU activations, where mid dim is $\max(64, \text{input dim}/2)$.
- **Decoder**: Two linear layers (latent dim $\rightarrow$ mid dim, mid dim $\rightarrow$ input dim) with ReLU activations reconstruct the input.

CNN The `CNN` model uses a convolutional neural network for time-series forecasting:

- **CNN**: Two 1D convolutional layers process the time-series input (`[batch, 168, 4]`):
  - Layer 1: 4 $\rightarrow$ 32 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Layer 2: 32 $\rightarrow$ 32 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).

  - Output: `[batch, 42, 32]` (sequence length reduced to 168/4 = 42 after two pooling layers).
- **Flatten**: Flattens the CNN output to a vector of size 32 $\times$ 42 = 1344 (`[batch, 1344]`).
- **Output Layers**: Two linear layers (1344 $\rightarrow$ 128, 128 $\rightarrow$ 24) with ReLU activation produce the 24-step forecast (`[batch, 24]`).

DualFCNN The `DualFCNN` model processes time-series and covariate data with positional encoding and a feedforward neural network decoder:

- **Primary Use Embedding**: Maps 16 primary use categories to a 16-dimensional vector (`[batch, seq_len, 16]`).
- **Time-Series Projection**: A linear layer (4 $\rightarrow$ 64) projects the time-series input (`[batch, 168, 4]`) to a higher-dimensional space (`[batch, 168, 64]`).
- **Positional Encoding**: Adds positional information (hidden size 64, max length 168) to the time-series input (`[batch, 168, 64]`).
- **Time-Series ANN**: A feedforward neural network processes the time-series input:
  - Flatten: Converts `[batch, 168, 64]` to `[batch, 168 × 64]`.
  - Hidden Layers: Two linear layers (10752 $\rightarrow$ 64, 64 $\rightarrow$ 64) with ReLU activations.
  - Output: `[batch, 64]`.
- **Covariate ANN**: A feedforward neural network processes the concatenated covariate and embedding input (`[batch, 168, 17]`):
  - Flatten: Converts `[batch, 168, 17]` to `[batch, 168 × 17]`.
  - Hidden Layers: Two linear layers (2856 $\rightarrow$ 64, 64 $\rightarrow$ 64) with ReLU activations.
  - Output: `[batch, 64]`.
- **Decoder**: Three linear layers (128 $\rightarrow$ 64, 64 $\rightarrow$ 32, 32 $\rightarrow$ 24) with ReLU and dropout (0.2) produce the 24-step forecast (`[batch, 24]`).

DualCNNANN The `DualCNNANN` model combines convolutional and feedforward neural networks with positional encoding:

- **Primary Use Embedding**: Maps 16 primary use categories to a 16-dimensional vector (`[batch, seq_len, 16]`).
- **Time-Series Projection**: A linear layer (4 $\rightarrow$ 64) projects the time-series input (`[batch, 168, 4]`) to a higher-dimensional space (`[batch, 168, 64]`).
- **Positional Encoding**: Adds positional information (hidden size 64, max length 168) to the time-series input (`[batch, 168, 64]`).
- **Time-Series CNN**: Two 1D convolutional layers process the projected time-series input:
  - Layer 1: 64 $\rightarrow$ 64 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Layer 2: 64 $\rightarrow$ 128 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Output: `[batch, 42, 128]` (sequence length reduced to 168/4 = 42).
- **Covariate CNN**: Two 1D convolutional layers process the concatenated covariate and embedding input (`[batch, 168, 17]`):
  - Layer 1: 17 $\rightarrow$ 32 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Layer 2: 32 $\rightarrow$ 64 channels, kernel size 5, padding 2, ReLU, max-pooling (kernel size 2).
  - Output: `[batch, 42, 64]`.
- **ANN**: A feedforward neural network processes the concatenated CNN outputs (`[batch, 42, 192]`):
  - Flatten: Converts `[batch, 42, 192]` to `[batch, 42 × 192]`.
  - Hidden Layers: Two linear layers (8064 $\rightarrow$ 64, 64 $\rightarrow$ 64) with ReLU activations.
  - Output: `[batch, 64]`.
- **Decoder**: Three linear layers (64 $\rightarrow$ 64, 64 $\rightarrow$ 32, 32 $\rightarrow$ 24) with ReLU and dropout (0.2) produce the 24-step forecast (`[batch, 24]`).

AutoARIMA

The ARIMA model is a traditional statistical model for time-series forecasting, implemented using the `auto_arima` function from the `pmdarima` library:

- **Configuration**: The model is fitted with `seasonal=False`, m = 168, `stepwise=True`.

Computation Overhead of Client Sampling

This appendix analyzes the *additional computation overhead* introduced by client sampling strategies in federated learning, beyond the standard local training and aggregation costs of FedAvg. We focus exclusively on the computation required to *select clients* in each communication round.

Notation. Let N be the total number of clients, m the number of selected clients per round, d the candidate pool size in power-of-choice sampling, F the cost of a single forward pass of the global model on one mini-batch, and R the total number of communication rounds.

Assumptions. We assume:

- Local training and server-side aggregation costs are identical across sampling strategies and are excluded.
- Power-of-choice sampling requires evaluating the global model on one local mini-batch for each candidate client.
- Difficulty-aware sampling reuses loss values obtained during standard local training and incurs no extra model evaluations.
- Client statistics are stored as scalar values across rounds.

Power-of-Choice Sampling

In power-of-choice (PoC) sampling, each round proceeds by: (i) sampling a candidate set of size d, (ii) evaluating the global model on one mini-batch at each candidate client, and (iii) selecting the top-m clients based on estimated loss.

Sampling candidates costs $O(d)$. Loss evaluation incurs $O(d \cdot F)$ due to d forward passes of the global model. Ranking candidates requires $O(d \log d)$ time.

The total computation overhead per round is therefore

$$O(d \cdot F + d \log d). \tag{1}$$

This overhead arises from additional inference-only evaluations on candidate clients that do not necessarily participate in local training.

Difficulty-Aware Sampling

In difficulty-aware sampling (DAS), each client maintains a scalar difficulty score updated only when it participates in local training. Client selection is performed by sampling from a distribution defined by these stored scores.

Updating difficulty statistics for participating clients costs $O(m)$. Computing sampling probabilities over all clients costs $O(N)$. Sampling m clients adds an $O(m)$ cost.

Hence, the total overhead per round is

$$O(N + m), \tag{2}$$

which simplifies to $O(N)$ since typically $N \gg m$.

DAS requires no additional forward or backward passes of the model, as all loss values are obtained during standard local optimization.

Memory and Computation Summary

Power-of-choice sampling does not maintain persistent client-level state, but it incurs repeated inference-time evaluations of the global model on candidate clients at every round, resulting in an additional computation cost of $O(d \cdot F)$ per round.

In contrast, difficulty-aware sampling maintains $O(N)$ client-level scalar statistics, which is negligible compared to model parameters and client data. Its per-round overhead consists only of $O(N)$ scalar operations and requires no additional forward or backward passes of the model.

Overall, difficulty-aware sampling is both memory-efficient and computationally lightweight, and therefore scales more favorably than power-of-choice sampling to large client populations.

Evaluation Metrics

Normalized Root Mean Square Error (NRMSE) The NRMSE is the primary metric used in this study as it penalizes large deviations more heavily than smaller errors, making it suitable for variance-sensitive forecasting tasks. For a building with M days of load time series and 24-hour forecasts:

$$\text{NRMSE} := 100 \times \frac{\sqrt{\frac{1}{24M} \sum_{j=1}^{M} \sum_{i=1}^{24} (y_{j,i} - \hat{y}_{j,i})^2}}{\bar{y}}$$

where $y_{j,i}$ is the actual load at hour i on day j, $\hat{y}_{j,i}$ is the predicted load, and $\bar{y} = \frac{1}{24M} \sum_{j=1}^{M} \sum_{i=1}^{24} y_{j,i}$ is the mean of the observed loads.

Symmetric Mean Absolute Percentage Error (SMAPE) SMAPE is a scale-independent metric that symmetrically measures percentage errors, avoiding the asymmetry issues of traditional MAPE. It is bounded between 0% and 200%:

$$\text{SMAPE} := 100 \times \frac{1}{24M} \sum_{j=1}^{M} \sum_{i=1}^{24} \frac{|y_{j,i} - \hat{y}_{j,i}|}{(|y_{j,i}| + |\hat{y}_{j,i}|)/2}$$