Diffusion Model Based Probabilistic Day-ahead Load Forecasting

Ding Lin, Graduate Student Member, IEEE, Han Guo, Graduate Student Member, IEEE, Jianhui Wang, Fellow, IEEE

Abstract-Accurate probabilistic load forecasting is crucial for maintaining the safety and stability of power systems. However, the mainstream approach, multi-step prediction, must be improved by cumulative errors and latency issues, which limits its effectiveness in probabilistic day-ahead load forecasting (PDALF). To overcome these challenges, we introduce DALNet, a novel denoising diffusion model designed to generate load curves rather than relying on direct prediction. By shifting the focus to curve generation, DALNet captures the complex distribution of actual load time-series data under specific conditions with greater fidelity. To further enhance DALNet, we propose the temporal multi-scale attention block (TMSAB), a mechanism designed to integrate both positional and temporal information for improved forecasting precision. Furthermore, we utilize kernel density estimation (KDE) to reconstruct the distribution of generated load curves and employ KL divergence to compare them with the actual data distribution. Experimental results demonstrate that DALNet excels in load forecasting accuracy and offers a novel perspective for other predictive tasks within power systems.

Index Terms—Diffusion model, probabilistic forecasting, attention mechanism.

I. INTRODUCTION

AY-ahead electrical load forecasting plays a key role in balancing electricity supply and demand, offering valuable insights for planning, construction, and operational decision-making in power systems. With the increasing integration of renewable energy, flexible resources, and advancements in electricity markets and demand response mechanisms, load forecasting has become more complex due to heightened volatility and uncertainty. In this evolving landscape, accurate and reliable probabilistic day-ahead load forecasting serves as a crucial tool for mitigating uncertainties and ensuring safe operational control within the power system.

Currently, day-ahead forecasting can be broadly categorized into three methods. The first method is rolling forecasting, the second involves building individual models for each time point, and the third is multi-step prediction. The earliest methods for rolling forecasts employed statistical models like the autoregressive integrated moving average model [1] and the multiple linear regression model [2]. These linear models often need to catch up in capturing nonlinear relationships accurately. To address this issue, researchers have explored machine learning models such as support vector machines [3] and random forests [4]. Despite their advantages, traditional machine learning algorithms sometimes need help modeling complex nonlinear relationships and managing large datasets effectively. Consequently, neural networks have gained significant attention due to their superior performance in handling

complex nonlinear mappings. For instance, Bi-directional long short-term memory (Bi-LSTM) [5] and feedback neural network [6] have been employed for rolling forecasts.

Rolling forecasts often focus on point predictions and face the issue of error accumulation. Researchers have started modeling each time step individually for day-ahead load forecasting to address these challenges. [7] developed 24 multiple linear regression models, one for each hour of the day, for day-head forecasting. In addition, scholars often use neural networks to implement this method, such as the attention mechanism [8]. Furthermore, the second day-ahead load forecasting method can be easily extended to probabilistic forecasting methods through techniques like quantile regression (OR) [9] and KDE [10]. Y. Wang et al. [11] extended LSTMbased point forecasting to the quantile regression LSTM to handle the non-stationary and stochastic features of individual consumers. Taking the uncertainty of low-voltage load data into account to improve the accuracy of the probabilistic forecasting, Z. Cao et al. [12] proposed a hybrid ensemble learning model based on the deep belief network. X. Liu et al. [13] proposed an ordinary differential equation network combined with QR to capture the uncertainties. M. Sun et al. [14] proposed a probabilistic day-ahead net load forecasting method that combined Bayesian theory and LSTM to capture the epistemic and aleatoric uncertainty of load data.

Although the second method can avoid the accumulation of prediction errors, independent modeling of each time point needs to pay attention to the relationships between different times of the day, which is a significant limitation for capturing intra-day load patterns with apparent regularities. The third method, multi-step prediction, has emerged to address this issue and has become the mainstream forecasting approach. By utilizing a transformer to capture the periodicity in load data, B. Jiang et al. [15] achieved multi-step forecasting. To improve the accuracy of day-ahead load forecasting of the Transformer, K. Qu et al. [16] proposed a novel model excelling in predictions for special days such as weekends and holidays. For day-ahead probabilistic load forecasting, [17] first utilized ensemble learning for point forecasting, followed by Markov Chain Monte Carlo methods to generate the distribution of the forecasted day and achieve probabilistic forecasting. [18] leveraged a hybrid approach, combining Convolutional Neural Networks and Gated Recurrent Units for load forecasting. followed by QR to achieve probabilistic prediction. Gaussian Mixture Model [19] and KDE [20] are also used to achieve probabilistic forecasting.

Although researchers widely apply multi-step prediction

methods, these methods inherently experience latency issues, which become particularly pronounced when dealing with rapidly changing data [21]. Therefore, new prediction methods are needed to address the shortcomings of error accumulation, lack of consideration for intra-day correlations, and the latency issue in the three methods above. To this end, we propose a prediction method based on conditional diffusion probabilistic models (DDPM), a generative model, to generate load data for day-ahead forecasting directly. DDPMs have demonstrated remarkable abilities in learning complex, high-dimensional data distributions and generating realistic samples. By adding Gaussian noise in a forward process and then learning to reverse it, DDPMs can effectively recover the original data distribution, making them highly effective for generative tasks. The decision to use DDPMs stems from their unique capabilities. Unlike rolling forecasting, DDPMs can generate an entire curve without depending on the generated value from the previous time step, which avoids error accumulation. Additionally, by considering the interconnections between intra-day loads, DDPMs establish a probabilistic distribution for the entire day to address intra-day dependency issues. Furthermore, since this approach generates curves that share the same distribution as the target day rather than directly predicting values, it eliminates latency concerns. Recent work [22] proposed a Seq2Seq diffusion-based approach for probabilistic load forecasting to quantify both epistemic and aleatoric uncertainty. The method assumes that aleatoric uncertainty in load forecasting follows a Cauchy distribution. While this heavy-tailed distribution provides robustness against outliers, it is a fixed assumption and may not always align with realworld data distributions, which could exhibit mixed or nonparametric characteristics.

In this paper, rather than imposing strict parametric distribution assumptions, we propose a diffusion-based generative framework DALNet, which integrates neural networks such as LSTM and our developed TMSAB mechanism to learn the underlying distribution of load data directly. The primary contributions of our approach are summarized below:

- The paper introduces a novel diffusion-based framework tailored for PDALF. By progressively noising and denoising time-series data in a Markov chain, the model can effectively learn the underlying distribution of load curves and generate forecasts in a probabilistic manner.
- 2) We design a new denoising network, DALNet, specifically for load data, in which we develop an original attention mechanism called TMSAB. This mechanism simultaneously considers positional and temporal information within the sequence to more accurately capture the correlations between load data.
- 3) We reconstruct the distribution of the real and generated load curves using KDE and compare them using KL divergence. Experimental results show that the proposed model, DALNet, can effectively fit the original data distribution and outperforms other benchmarks.

The rest of this paper is structured as follows: Section II covers the concepts of DDPM. Section III provides the technical details of the denoising network, DALNet. Section

IV presents the case studies. Finally, Section V concludes the paper.

II. DIFFUSION-BASED FORECASTING MODEL

In this section, we explain the principles of denoising diffusion models. The overview of the DDPM is elaborated in Fig. 1. Given samples from a data distribution $q(\mathbf{x}_0)$, DDPM is an unconditional generative model designed to learn a model distribution $p_{\theta}(\mathbf{x}_0)$ that approximates $q(\mathbf{x}_0)$ and is easy to sample. The diffusion model is designed to reconstruct this distribution with two steps: the forward process and the reverse process. The forward process involves progressively adding Gaussian noise to the original sample, which eventually transforms it into pure Gaussian noise. Conversely, the reverse process gradually removes the Gaussian noise and finally reconstructs the original sample from the noisy data. Without loss of generality, we use \mathbf{x}_0 to represent the original load data, and $\mathbf{x}_1, ..., \mathbf{x}_T$ to represent the data with added noise, with \mathbf{x}_T denoting pure Gaussian noise.

A. Forward Diffusion Process

Diffusion models have gained significant attention for their ability to generate high-dimensional data with excellent training stability compared to other generative models. Our approach begins by applying forward diffusion to convert real load curves into noisy samples. We then use reverse diffusion to reconstruct the load time series from this perturbed noise. The forward and reverse processes are modeled as Markov chains, with the reverse Gaussian transitions learned via a deep denoising neural network. Gaussian noise is incrementally added to the original load time-series data \mathbf{x}_0 during the forward process. After T diffusion steps, the original distribution, $q(\mathbf{x}_0)$ is transformed into a standard Gaussian distribution $q(\mathbf{x}_T)$. This process of adding noise can be described as a fixed Markov chain:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{1}$$

where $\mathbf{x}_1, ..., \mathbf{x}_T$ can be viewed as latent variables representing the intermediate states that result from perturbing the actual load curves with Gaussian noise at each step t. The term $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ represents the forward Markov transition, specifying the mean and variance of the Gaussian noise added to \mathbf{x}_{t-1} :

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}),$$
 (2)

where $\{\beta_1,...,\beta_T\}$ is an increasing variance schedule with $\beta_t \in (0,1)$ that represents the noise level at forward step t. Unlike typical latent variable models such as the variational autoencoder (VAE), the approximate posterior distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ in diffusion models is not trainable but is fixed to follow the aforementioned Gaussian transition process. Following the T-step diffusion process, a certain load curve is ultimately transformed into a straightforward Gaussian noise \mathbf{x}_T , which facilitates easier sampling and manipulation.

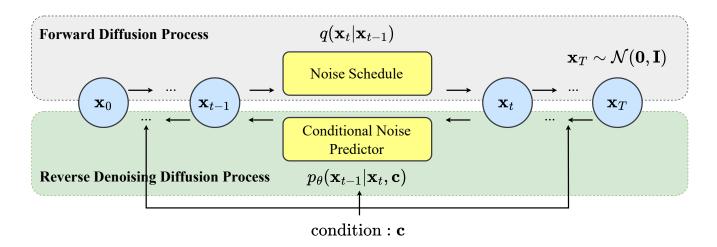


Fig. 1: Overview of the conditional diffusion probabilistic models.

Let $\alpha_t = \prod_{n=1}^t (1-\beta_n)$, a particular property of the forward process is that the distribution of \mathbf{x}_t given \mathbf{x}_0 has a close form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbf{I}). \tag{3}$$

Using the reparameteriztioin trick and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ as a sampled noise, (3) can be expressed as:

$$\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 - \alpha_t} \boldsymbol{\epsilon}. \tag{4}$$

Based on (4), we can directly sample x_t at any arbitrary noise level t, instead of computing the forward process step by step. The detailed proof of (3) and (4) can be found in [23].

B. Reverse Denoising Process

In contrast to the noise addition in the forward process, the reverse process progressively removes noise from the initial standard Gaussian noise \mathbf{x}_T to recover the original load curve \mathbf{x}_0 . The reverse process also follows a Markov chain with learnable Gaussian transitions starting from $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$. Inspired by non-equilibrium thermodynamics, if we know the step-wise Gaussian noise applied in the forward process, we can restore the real load curve distribution through a series of iterative denoising steps. Such reverse process can be represented as follows.

$$p_{\theta}(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{5}$$

where $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $p_{\theta}(\cdot)$ represents the reverse Markov transition. Based on the statistical characteristics of the continuous diffusion process outlined in [24], when the added Gaussian noise is sufficiently small, the denoising transition $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will resemble the functional form of the forward transition $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. Therefore, p_{θ} can represent a learnable Gaussian transition, which can be approximated by a neural network with θ representing the network parameters. Moreover, the transition between two adjacent latent variables is indicated by:

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t)), \tag{6}$$

with shared parameters θ , Here, we adopt the same parameterization of $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ as in [24] due to its demonstrated effectiveness in image generation:

$$\mu_{\theta}(\mathbf{x}_{t}, t) = \frac{1}{\sqrt{1 - \beta_{t}}} \left(\mathbf{x}_{t} - \frac{\beta_{t}}{\sqrt{1 - \alpha_{t}}} \epsilon_{\theta}(\mathbf{x}_{t}, t) \right)$$
(7a)

$$\Sigma_{\theta}(\mathbf{x}_t, t) = \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \beta_t, \tag{7b}$$

where $\epsilon_{\theta}(\cdot)$ is a trainable denoising function determining the amount of noise to remove at each denoising step. Our goal is to design an efficient method to learn how to sample from $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and ultimately derive the load curve distribution $p_{\theta}(\mathbf{x}_0)$.

C. Training Objective

Instead of optimizing (6) directly due to its complexity, we approximate the real distribution of load curves by maximizing their log-likelihoods. Furthermore, directly solving the $log p_{\theta}(\mathbf{x}_0)$ is impractical and difficult to compute explicitly. Hence, we typically opt to optimize its Evidence Lower Bound (ELBO) instead, which can be described as follows:

$$\log p_{\theta}(\mathbf{x}_{0}) = \log \int p_{\theta}(\mathbf{x}_{0:T}) d(\mathbf{x}_{1:T})$$

$$= \log \int \frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_{0})} q(\mathbf{x}_{1:T}|\mathbf{x}_{0}) d(\mathbf{x}_{1:T})$$

$$= \log \left(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_{0})} \left[\frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_{0})} \right] \right)$$

$$\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_{0})} \left[\log \frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_{0})} \right].$$
(8)

The final inequality in (9) is derived from Jensen's inequality. It is important to note that $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ is the posterior distribution of $p_{\theta}(\mathbf{x}_0)$, defined by a sequence of latent variables $\mathbf{x}_1,...,\mathbf{x}_T$ in the forward process, and it can be readily computed using (2). Consequently, maximizing $\log p_{\theta}(\mathbf{x}_{0:T})$ is equivalent to minimizing its negative ELBO. This negative ELBO can be decomposed into T+1 tractable items:

$$ELBO = -(\mathcal{L}_0 + \Sigma_T^{t=2} \mathcal{L}_{t-1} + \mathcal{L}_T), \tag{9}$$

the exact expressions of \mathcal{L}_0 , \mathcal{L}_{t-1} , and \mathcal{L}_T are illustrated as follows:

$$\mathcal{L}_{0} = -\mathbb{E}_{q(\mathbf{x}_{1}|\mathbf{x}_{0})}[\log p_{\theta}(\mathbf{x}_{0}|\mathbf{x}_{1})]$$

$$\mathcal{L}_{t-1} = \mathbb{E}_{q(\mathbf{x}_{t}|\mathbf{x}_{0})}[\mathcal{D}_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_{t},\mathbf{x}_{0})||p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_{t}))]$$
(10b)

$$\mathcal{L}_T = \mathcal{D}_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p_{\theta}(\mathbf{x}_T)), \tag{10c}$$

where \mathcal{D}_{KL} is the KL Divergence, and $\mathcal{L}_0, \mathcal{L}_{t-1}, \mathcal{L}_T$ are referred to as the reconstruction term, the consistency term, and the prior matching term, respectively. The term \mathcal{L}_0 predicts the log probability of the original data sample given the first-step latent. Additionally, \mathcal{L}_0 is a special case of \mathcal{L}_{t-1} . When t = 1, \mathcal{L}_{t-1} transforms into \mathcal{L}_0 . \mathcal{L}_{t-1} aims to ensure consistency in the distribution at x_t for both the forward and backward processes, which means that each denoising step from a noisier image should correspond to the appropriate noising step from a cleaner image at every intermediate timestep. The term \mathcal{L}_T does not require optimization since it has no trainable parameters. Moreover, KL Divergence is used to measure the difference between two distributions. The smaller the difference, the smaller the value. When the two distributions are identical, the KL Divergence is 0. Given our assumption of a sufficiently large T such that the final distribution is Gaussian, this term effectively becomes zero. The KL Divergence mathematically represents this relationship. The term $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$ can be analytically calculated as follows:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

$$= \mathcal{N}(\mathbf{x}_{t-1}; \widetilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \widetilde{\beta}_t \mathbf{I}).$$
(11)

The first line of (11) utilizes the Bayesian rule, which makes the conditional probability tractable since the right-hand side of (11) involves three known Gaussian distributions. The mean and variance defined in (11) are as follows:

$$\widetilde{\mu}_{t}(\mathbf{x}_{t}, \mathbf{x}_{0}) = \frac{\sqrt{\alpha_{t-1}}\beta_{t}}{1 - \alpha_{t}} \mathbf{x}_{0} + \frac{\sqrt{1 - \beta_{t-1}}(1 - \alpha_{t-1})}{1 - \alpha_{t}} \mathbf{x}_{t}$$

$$= \frac{1}{\sqrt{1 - \beta_{t}}} (\mathbf{x}_{t} - \frac{\beta_{t}}{\sqrt{1 - \alpha_{t}}} \boldsymbol{\epsilon}),$$

$$= \frac{1 - \alpha_{t-1}}{2} \mathbf{x}_{t} + \frac{1 - \alpha_{t-1}}{2} \mathbf{x}_{t}$$
(12)

 $\widetilde{\beta}_t = \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \beta_t. \tag{13}$

In (11), it should be noted that \mathbf{x}_0 can be derived from \mathbf{x}_t and $\boldsymbol{\epsilon}$ based on (4). Following the empirical simplification adopted in [24], we also fix $\Sigma_{\theta}(\mathbf{x}_t,t) = \widetilde{\beta}_t \mathbf{I}$ for $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which eliminates the need to learn the variance of the reverse transition. Consequently, \mathcal{L}_{t-1} reduces to calculating the KL divergence between the two Gaussian distributions defined in (6) and (11), which have different means but the same variance. Therefore, \mathcal{L}_{t-1} can be computed as follows:

$$\mathcal{L}_{t-1} = \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \left[\frac{1}{2\widetilde{\beta}_t} ||\widetilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_{\theta}(\mathbf{x}_t, t)||_2^2 \right], \quad (14)$$

which aims to predict the mean of the reverse transition. According to (12) and the method proposed in [24], an alternative parameterization for $\mu_{\theta}(\mathbf{x}_t, t)$ can be defined in (7a). This parameterization suggests a more efficient way for training the denoising network, specifically by directly

predicting the Gaussian noise ϵ added at step t instead of the mean $\widetilde{\mu}_t$ of the reverse transition. Consequently, \mathcal{L}_{t-1} can be reformulated as follows:

$$\mathcal{L}_{t-1} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t}[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}, t)||_2^2], \quad (15)$$

where t is uniformly sampled from [1,T], and $\epsilon_{\theta}(\cdot)$ denotes the denoising neural network. $\epsilon_{\theta}(\cdot)$ takes the perturbed \mathbf{x}_t defined in (4) as input and outputs the Gaussian noise prediction $\hat{\epsilon}$ at step t. After training the denoising network $\epsilon_{\theta}(\cdot)$ using (15), it can be used to incrementally reconstruct the load curves during the reverse denoising process by computing $\mu_{\theta}(\mathbf{x}_t,t)$ as given in (7a).

D. Conditional Diffusion Model

The standard DDPM is intended to generate images from pure white noise without any conditions, which is not suitable for our task of generating future load curves based on historical data. To address this, we introduce conditional DDPM, where the model is conditioned on past load information to produce the desired future load curves.

In the original DDPM, the reverse process $p_{\theta}(\mathbf{x}_{0:T})$ as outlined in (5) is employed to estimate the final data distribution $q(\mathbf{x}_0)$. For the conditional DDPM, this equation needs to be adjusted as follows:

$$p_{\theta}(\mathbf{x}_{0:T}|\mathbf{c}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}),$$
(16)

where c represents the condition. Similarly, (6) and (15) are revised as:

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_{t},\mathbf{c}) = \mathcal{N}(\mathbf{x}_{t-1};\mu_{\theta}(\mathbf{x}_{t},\mathbf{c},t),\widetilde{\beta}_{t}\mathbf{I})),$$
 (17)

$$\mathcal{L}_{t-1}^{cond} = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t}[||\boldsymbol{\epsilon} - \epsilon_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}, \mathbf{c}, t)||_2^2].$$
 (18)

To maximize the ELBO, we need to minimize \mathcal{L}_{t-1}^{cond} . Therefore, the optimization objective can be defined as follows:

$$\min_{\theta} \mathcal{L}(\theta) = \min_{\theta} \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t}[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1 - \alpha_t}\boldsymbol{\epsilon}, \mathbf{c}, t)||_2^2].$$
(19)

According to (19), we ultimately need to design a function $\epsilon_{\theta}(\cdot)$ to predict the noise. $\epsilon_{\theta}(\cdot)$ takes the original sample \mathbf{x}_0 , the condition \mathbf{c} , standard Gaussian noise ϵ , and the noise addition step t as inputs. The expectation calculation \mathbb{E} can be accomplished through uniform sampling. (19) represents the final implementation of DDPM, and the training procedure of conditional DDPM is presented in Algorithm 1.

Algorithm 1 Training of Conditional DDPMs

Require: Load curve datasets with conditions $q(\mathbf{x}_0|\mathbf{c})$.

Ensure: Denoising diffusion model $\epsilon_{\theta}(\cdot)$.

- 1: while θ has not converged do
- 2: $\mathbf{x}_0 \sim q(\mathbf{x}_0|\mathbf{c}), \ \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \text{Uniform}(\{1, ..., T\})$
- 3: Update the denoising network using gradient descent: $\nabla || \boldsymbol{\epsilon} \epsilon_{\theta} (\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{1 \alpha_t} \boldsymbol{\epsilon}, \mathbf{c}, t ||_2^2 \text{ using (18)}$
- 4: end while

Once a noise-predicting DDPM is trained, we can restore the original curves by progressively removing the noise through reverse. The algorithm for the reverse process is illustrated in Algorithm 2.

Algorithm 2 Sampling of Conditional DDPMs

Require: Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, condition \mathbf{c} .

Ensure: Generated load curves.

1: **for** t = T, ..., 1 **do**

2: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

3: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\epsilon_{\theta}(\mathbf{x}_t, \mathbf{c}, t)) + \sqrt{\widetilde{\beta}_t}\mathbf{z} \text{ according to both (6) and (7a).}$

4: end for

III. DETAILS OF DENOISING ARCHITECTURE

According to (19), we need to design a denoising network $\epsilon_{\theta}(\cdot)$ to predict the noise and progressively remove the predicted noise to generate the day-ahead load curve. Therefore, designing $\epsilon_{\theta}(\cdot)$ that inputs historical load data is crucial for constructing an effective diffusion model. This section will illustrate how to design a denoising network DALNet suitable for load time series.

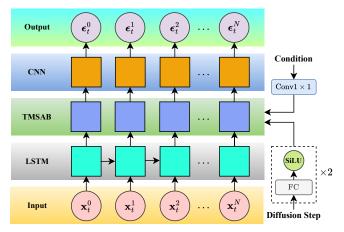


Fig. 2: Structure of the proposed denoising network DALNet.

A. Overview of Denoising Network

Fig. 2 depicts the overall structure of the denoising model DALNet. Given that our load data is a time series with distinct temporal characteristics, it is intuitive to design a neural network capable of extracting these features as the primary framework for $\epsilon_{\theta}(\cdot)$. This architecture aims to preserve the temporal associations during the reverse generation process. Therefore, we utilized LSTM, a classic neural network for processing time series, and our improved attention mechanisms, TMSAB, which have demonstrated advantages in handling textual data. Given an input sample x_0 , we can obtain the noise sample $\mathbf{x}_t \in \mathbb{R}^{N \times 1}$ using (4), where N presents the length of time-series x_t . The output of LSTM can be expressed as $\mathbf{h}_t = LSTM(\mathbf{x}_t)$, with $\mathbf{h}_t \in \mathbb{R}^{N \times H}$. H is the dimension of the hidden state vector. Another reason for choosing LSTM is that it effectively alleviates the vanishing gradient problem and offers enhanced nonlinearity due to its multiple gate functions. Details about LSTM can be found in [25]. The output of the LSTM, combined with the diffusion step embedding and condition embedding, serves as the input for the TMSAB, which not only captures temporal correlations in the load curve but also extracts positional information within the sequence. The output of the attention mechanism is fed into 1D-convolutional layers, which generate the final noise prediction result $\epsilon_t \in \mathbb{R}^{N \times 1}$.

B. Temporal Multi-scale Attention Mechanism

In [26], a multi-head attention mechanism is introduced. This approach computes attention scores through linear transformations applied to the queries, keys, and values (Q, K, V). Assuming the dimension of the multi-head attention input $\mathbf{x} = \{x_1, ..., x_n, ..., x_N\}$ is $\mathbb{R}^{N \times H}$, where N represents the sequence length and the dimension of x_n is $1 \times H$. First, by focusing on one of the heads, the attention mechanism applies a linear transformation to each element in the sequence to generate $Q=\{q_1,...,q_n,...,q_N\}$ and $K=\{k_1,...,k_n,...,k_N\}$: $Q=\mathbf{x}*\mathbf{w}_Q, K=\mathbf{x}*\mathbf{w}_K.$ Both Q and K have the same dimension of $\mathbb{R}^{N\times M}$, and M denotes the dimension of the attention model, which means the element in the sequence has its corresponding q and k. By multiplying the q_i and k_i , we can derive an attention score a_{ij} , representing the relationship between the i-th element and the j-th element in the sequence. In a similar way, by multiplying Q and K, we can obtain an attention map **A** with the dimension of $\mathbb{R}^{N \times N}$.

$$\mathbf{A} = \operatorname{softmax}(\frac{QK^T}{\sqrt{M}}). \tag{20}$$

The diagonal elements of the attention map indicate the self-attention scores, while the other elements represent the mutual attention scores between different elements. This type of attention mechanism is also called global attention, with the diagram of its attention map shown in Fig. 3(a). When considering multiple heads, we obtain multiple attention maps.

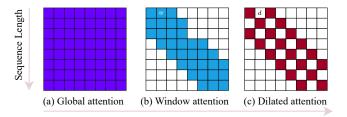


Fig. 3: Illustration of three attention maps. (a): Global attention map; (b): Window attention map with window size w=2; (c) Dilated window attention map with dilation d=1.

After generating the attention map, it can be transformed into the output O of one head through a linear transformation matrix $V: V = \mathbf{x} * \mathbf{w}_V$ and $O = \operatorname{softmax}(\frac{QK^T}{\sqrt{M}})V$. For multihead attention, we stack outputs of all heads and also use a linear transformation:

Multihead =
$$[O_1, O_2, ... O_h] * \mathbf{w}_O,$$
 (21)

where h is the number of the attention heads.

A critical factor in its effectiveness is that the self-attention component enables the model to grasp contextual information across the entire sequence. Despite its power, the selfattention mechanism has significant drawbacks: its memory and computational requirements escalate quadratically with the length of the sequence, making it inefficient or prohibitively expensive to handle long sequences. What is more, Algorithm 2 reveals that the sampling process in the diffusion model requires iteration from T down to 1. When T is large and the sequence is lengthy, using a global attention mechanism significantly increases the computation time needed. Besides the computational challenge, [27] highlights that the diversity of multi-head attention needs to be improved. Often, the attention heads are highly repetitive and need to focus on distinct representation subspaces as intended.

Therefore, to address the issues above, we introduced the dilated window attention mechanism as described in [28]. According to (20), we know that for a sequence \mathbf{x} each of its elements computes an attention score a_{ij} with all other elements, while window attention refers to the mechanism where, for the i-th element in a sequence \mathbf{x} , the attention score is calculated only with a few neighboring elements. A diagram of the window attention map can be found in Fig. 3(b). In this way, it reduces the computational cost and places greater emphasis on the weights of the elements adjacent to the i-th element. In addition, to further increase the diversity of the attention map, we also incorporated dilated window attention, which can be found in Fig. 3(c).

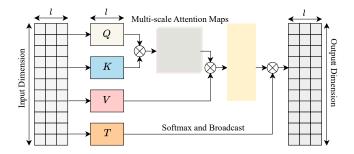


Fig. 4: Visualization of the TMSAB with the sequence length l.

Although the multi-scale attention map addresses the issue of highly repetitive attention heads, it tends to focus more on positional information within the sequence while neglecting temporal information [29]. To resolve this issue, we introduced a temporal embedding T and proposed a new attention mechanism called TMSAB, with the visualization of TMSAB illustrated in Fig. 4. Like the method of generating Q, K, V in global attention, we first apply a linear transformation $T = \mathbf{x} * \mathbf{w}_T$ to the input data to convert it into T. Then, we use an activation function $\tanh(\cdot)$ to mitigate the vanishing gradient problem, followed by a transformation into an $l \times 1$, where l is the sequence length. Finally, a temporal attention vector is generated using softmax, and broadcasting ensures it meets the required dimensions. The complete expression for the temporal attention T is as follows:

$$\mathbf{T} = \operatorname{softmax}(\tanh(\mathbf{x} * \mathbf{w}_T) * \mathbf{w}_I). \tag{22}$$

By designing TMSAB, we address the issue of highly repetitive attention maps in global attention and resolve the problem of multi-scale attention focusing solely on positional relationships. Therefore, introducing TMSAB is well-suited for load data, which is inherently a time series.

C. Diffusion Step and Conditionality Embedding

According to Algorithm 1 and 2, the denoising diffusion model $\epsilon_{\theta}(\cdot)$ has three inputs: the sample \mathbf{x}_t , the diffusion step t, and the condition \mathbf{c} . Notably, t plays a crucial role in $\epsilon_{\theta}(\cdot)$ as different values of t result in varying noise intensities. To incorporate the time step t into the model, which is akin to adding positional information, position embedding as introduced by [26] has been employed, which involves converting t into a vector comprising sine and cosine functions at different frequencies. A vector dimension of 64 is chosen for embedding the time step t, which can be represented as follows:

$$t_{\text{embedding}} = \left[\sin\left(10^{\frac{0\times4}{31}}t\right), \dots, \sin\left(10^{\frac{31\times4}{31}}t\right), \\ \cos\left(10^{\frac{0\times4}{31}}t\right), \dots, \cos\left(10^{\frac{31\times4}{31}}t\right)\right].$$
 (23)

Once the embedding is generated, it passes through two fully connected layers, each utilizing a SiLU activate function: $SiLU(FC(SiLU(FC(t_{embedding}))))$. The choice of SiLU is due to its ability to alleviate the vanishing gradient problem [30].

Theoretically, for condition embedding, day-ahead load generation can utilize various conditions, such as the predicted highest and lowest loads for the day ahead, and other meteorological forecast data like temperature and humidity. However, we opt not to use these conditions because they require prior predictions, and any inaccuracies in the load and weather forecasts would lead to inaccuracies in the generated load curve and results in cumulative errors. We use the previous day's load as the condition to avoid using forecasted values inspired by [31]. Consequently, the dimensions of the condition should be the same as those of the generated load curve. This condition vector \mathbf{c} is processed through multiple 1×1 convolutions to meet the dimensional requirements.

D. Noise Schedule

According to Algorithm 2, the Gaussian noise subtracted during the sampling process varies in intensity, with larger values of $\frac{\beta_t}{\sqrt{1-\alpha_t}}$ indicating more substantial subtracted noise. The selection of β_t is crucial for the denoising model's ability to generate accurate load curves. Based on [32], we employ a quadratic schedule because it allows for smoothly adding noise incrementally, which enhances the generative ability of the diffusion model:

$$\beta_t = \left(\frac{T-t}{T-1}\sqrt{\beta_1} + \frac{t-1}{T-1}\sqrt{\beta_T}\right)^2.$$
 (24)

We follow the standard setting: $\beta_1 = 0.0001$ and $\beta_T = 0.5$. Based on this noise schedule, the noise coefficients $\sqrt{\alpha_t}$ and $\sqrt{1-\alpha_t}$ in (4) can be calculated. As t increases, $\sqrt{\alpha_t}$ gradually decreases while $\sqrt{1-\alpha_t}$ increases. According to (4), $\sqrt{\alpha_t}$ is the coefficient for the original sample \mathbf{x}_0 , and $\sqrt{1-\alpha_t}$ is the coefficient for the noise ϵ , indicating that as t increases, the original sample progressively transforms into a noisy sample, with the degree of noise increasing. Furthermore, the original sample completely transforms into Gaussian noise when t=T, $\sqrt{\alpha_t}=0$. This perspective also indicates that the quadratic noise schedule is reasonable.

IV. CASE STUDY

This section will elaborate on the dataset, evaluation metrics, benchmarks, and experimental results.

A. Data and Evaluation Criteria Description

We utilized the GEFcom2014 dataset, which includes four subsets: load, wind power, solar power, and electricity price. For our study, we focused on the load data to evaluate the performance of the proposed diffusion model. This load dataset spans 3,560 days, covering the period from January 1, 2001, at 1:00 AM to October 1, 2010, at 00:00, with a data resolution of one hour. More detailed information about the data can be found in [33]. The dataset was divided into training and testing sets with a train-test split ratio of 8:2 to ensure a robust evaluation.

To evaluate the proposed model and other benchmarks, we employed Reliability, Sharpness, and Overall Score as the evaluation metrics in this paper. Reliability is the primary metric for evaluating the quality of probabilistic forecasting models. Reliability implies that, given a specified prediction interval nominal confidence (PINC) $100(1-\alpha)\%$, where α is the significance level, the prediction interval coverage probability (PICP) should be as close to the PINC as possible. PINC generally represents the theoretical probability of the predicted intervals (PIs). At the same time, PICP is the probability that the actual load values fall within the PIs, with their difference called Absolute Coverage Error (ACE), where a smaller absolute value of ACE indicates more accurate predictions.

Sharpness refers to the extent to which the predicted distribution closely matches the actual distribution. It is measured by the average width (AW) of the PIs, which can be determined by generating intervals for all samples and calculating their average.

The final metric is the overall score, which can be calculated as follows:

$$S_{t}^{(\alpha)} = \begin{cases} -2\alpha W_{t}^{(\alpha)} - 4(\hat{L}_{t}^{(\alpha)} - y_{i}), & \text{if } y_{i} < \hat{L}_{t}^{(\alpha)} \\ -2\alpha W_{t}^{(\alpha)}, & \text{if } y_{i} \in \hat{I}_{t}^{(\alpha)} \\ -2\alpha W_{t}^{(\alpha)} - 4(y_{i} - \hat{U}_{t}^{(\alpha)}), & \text{if } y_{i} > \hat{U}_{t}^{(\alpha)} \end{cases}$$
(25a)

$$\overline{S}_t^{(\alpha)} = \frac{1}{|N|} \sum_{t \in N} S_t^{(\alpha)},\tag{25b}$$

where W represents the width of the PI at time step $t,\,I$ denotes the PI, and L and U represent the lower and upper bounds of the PI, respectively. (25a) indicates that for a given time t if the actual load value falls within the PI, the absolute value of S should be minimal. If not, an additional penalty term is added. (25b) calculates the overall score by averaging the S across all PIs. According to (25), S is negative, and the smaller the absolute value, the better the forecasting performance.

B. Benchmarks

Multi-layer perceptron (MLP), LSTM, Transformer, and Bayesian neural network (BNN) are selected as benchmarks. MLP is chosen as the benchmark for modeling different hours.

Specifically, for this benchmark, we trained 24 MLPs, one for each hour, with the size of hidden layers chosen from $\{32,64,128\}$. LSTM, Transformer, and BNN are all used for direct multi-step prediction, meaning they simultaneously predict the load values for the next 24 hours. We did not choose to use the rolling prediction method as a benchmark because rolling prediction is more commonly used for point forecasting. LSTM, as a classic time series prediction model, is chosen as a benchmark with the hidden size chosen from $\{48,96,192\}$.

Similarly, the Transformer, which has demonstrated powerful natural language sequence processing capabilities, is also widely used in time series prediction tasks and is therefore chosen as a benchmark, with the number of encoder and decoder layers set to 2. The three methods mentioned above all implement quantile regression by introducing the pinball loss function to achieve probabilistic forecasting. For the quantile crossing problem, we resolved it using the naive rearrangement method as described in [34].

As a probabilistic model, BNN does not require quantile regression to achieve probabilistic forecasting. BNN introduces probability distributions to the neural network's weights, which allows different weights and biases to be sampled during each inference. By performing multiple samples, probabilistic forecasting can be achieved. Therefore, BNN is also chosen as a benchmark. We implemented the BNN using convolutional layers and fully connected layers, with the number of channels chosen from $\{32,64\}$ and the kernel size chosen from $\{(3,1),(2,1)\}$, respectively.

C. Results Analysis of Evaluation Criteria

TABLE I
THE EVALUATION CRITERIA RESULTS FOR DIFFERENT MODELS

PINC	Attention Type	ACE	AW	$\overline{\mathbf{S}}^{(oldsymbol{lpha})}$
	MLP	2.72%	0.2110	-0.1112
80%	LSTM	5.72%	0.1759	-0.0898
	Transformer	4.95%	0.2154	-0.1094
	BNN	4.26%	0.1814	-0.0950
	DALNet	2.04%	0.1534	-0.0682
	MLP	0.62%	0.2640	-0.0657
90%	LSTM	3.06%	0.2265	-0.0538
	Transformer	2.42%	0.2731	-0.0650
	BNN	2.34%	0.2322	-0.0541
	DALNet	2.01%	0.1941	-0.0449
	MLP	0.72%	0.3034	-0.0376
95%	LSTM	1.16%	0.2671	-0.0309
	Transformer	0.53%	0.3137	-0.0370
	BNN	1.70%	0.2798	-0.0354
	DALNet	1.15%	0.2172	-0.0277

Table I presents the ACE, AW, and Overall Score $\overline{S}^{(\alpha)}$ results of five different models at three different PINC. It can be observed that at 80% PINC, the results of the proposed model are better than all other benchmarks. The ACE result is 0.68% better than the second-best result (MLP). The AW

and $\overline{S}^{(\alpha)}$ results are superior to LSTM by 0.0225 and 0.0216, respectively, which represent the second-best results. Similar to the results obtained at 80% PINC, when the PINC is set at 90%, the proposed model generally outperforms the benchmarks across the three metrics, except for the ACE, where it lags behind MLP by 1.39%. LSTM continues to achieve the second-best results in both AW and $\overline{S}^{(\alpha)}$, trailing the proposed model by 0.0381 and 0.0089, respectively. For the PINC of 95%, the proposed model ranks third in the ACE metric, with Transformer and MLP outperforming the proposed model by 0.62% and 0.43%, respectively. These results are mainly due to the significantly higher AW values obtained by these two models, which are greater than those of the proposed model by 0.0965 and 0.0862, respectively. Aside from these two models, LSTM continues to perform consistently, achieves the secondbest results in AW and $\overline{S}^{(\alpha)}$, and trails the proposed model by 0.0499 and 0.0032, respectively.

Overall, the proposed model generally outperforms the benchmarks across the three metrics, particularly in AW and $\overline{S}^{(\alpha)}$, where it significantly surpasses other models. These results are because the diffusion model effectively captures the distribution of the original data and generates results that align well with the original data distribution. For Transformer and BNN, part of their poorer performance is due to the difficulty in training these models. The Transformer requires a large number of parameters to be trained, and even when the number of encoder and decoder layers is set small, there are still over 100,000 parameters to train. As for BNN's training process, it involves Bayesian inference, which has a slow convergence rate and high computational complexity. Given the small sample size in our dataset, it is reasonable that the results for BNN could be more outstanding.

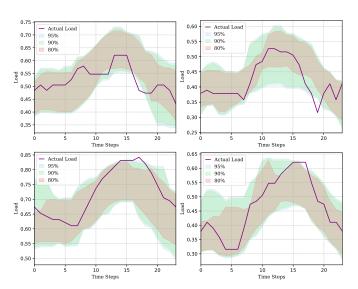


Fig. 5: Probabilistic forecasting results of the DALNet for the three loads under three different PINCs.

Fig. 5 presents the PIs generated by the diffusion models. It can be observed that the intervals produced by the diffusion model can capture the variation patterns of the intra-day load curves, as well as some minor fluctuations. Through Table

I and Fig. 5, it is evident that the proposed model shows considerable potential in PDALF tasks.

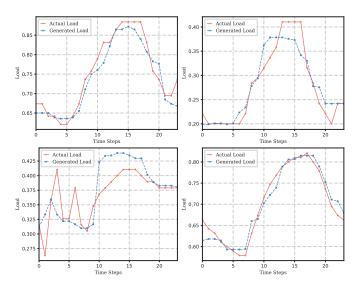


Fig. 6: Point forecasting results of the DALNet.

D. Evaluation of Generated Load Curves

Fig. 6 shows load curves generated by the diffusion model, obtained by averaging the generated curves. The MSE values are 0.0007, 0.0007, 0.0012, and 0.0004, respectively. Additionally, among the 4×24 time points in Fig. 6, the maximum deviation between the actual and generated curves is 0.0772, which is less than half of the AW value under the 95% PINC in Table I. It can be inferred that within the 95% PINC interval, this maximum deviation point also falls within the generated interval. Fig. 6 demonstrates that the generated curve accurately captures the overall trend of the actual load, which showcases the diffusion model's significant level of accuracy.

Moreover, we used KDE to construct the probability distributions of the generated and actual curves at 24-time points. We calculated the KL divergence between the distributions, illustrated in Fig 7. Among the 24 time steps, the maximum KL divergence is 0.0168 at the time step 1. According to [35], when the KL divergence is less than 0.05, the two distributions can be considered very similar. Therefore, through Fig. 6 and 7, we can see that the diffusion model not only generates curves but also captures the distribution of the generated curves. Compared to general prediction models such as LSTM and Transformer, the diffusion model demonstrates probabilistic capabilities that typical prediction models do not possess.

E. Comparison of Three Different Attention Mechanisms

Table II presents the numerical results obtained from three different attention mechanisms. When PINC is set to 80%, TMSAB achieved the best results across all three metrics, leading the second-best results by 1.14%, 0.0139, and 0.0029, respectively. With PINC at PINC = 90%, it only slightly lags behind Multi-scale attention in AW by 0.0002 while outperforming the other two attention mechanisms in the

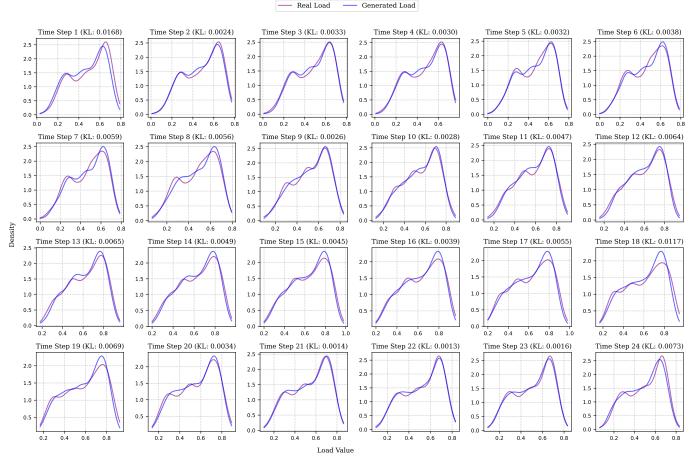


Fig. 7: KDE and KL divergence results across 24 time steps for the total test samples.

remaining metrics. At 95% PINC, the ACE metric is slightly lower than that of global attention by 0.07%.

Additionally, Fig. 8 displays the results of random samples at 95% PINC and the training loss for the three attention mechanisms. It is evident that TMSAB achieves superior results in both ACE and AW and demonstrates a faster convergence rate compared to the other two attention mechanisms.

TABLE II
THE EVALUATION CRITERIA RESULTS FOR DIFFERENT
ATTENTION MECHANISMS

PINC	Attention Type	ACE	AW	$\overline{\mathbf{S}}^{(oldsymbol{lpha})}$
80%	Global	3.32%	0.1559	-0.0711
	Multi-scale	3.18%	0.1673	-0.0736
	TMSAB	2.04%	0.1534	-0.0682
90%	Global	2.15%	0.2008	-0.0481
	Multi-scale	2.33%	0.1939	-0.0465
	TMSAB	2.01%	0.1941	-0.0449
95%	Global	1.22%	0.2196	-0.0282
	Multi-scale	1.72%	0.2229	-0.0293
	TMSAB	1.15%	0.2172	-0.0277

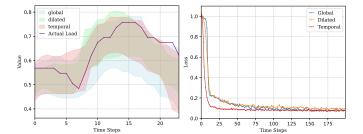


Fig. 8: Left: Probabilistic forecasting results of three different attention mechanisms under 95% PINC; Right: Training loss of different attention mechanisms.

V. CONCLUSION

In this paper, we introduce DALNet, a novel denoising diffusion model designed to achieve PDALF. To enhance DALNet, we developed an attention block, TMSAB, based on multi-scale attention, which integrates both positional and temporal information within the sequence. Experimental data demonstrate that DALNet effectively approximates the complex distribution of real load time-series data, with the inclusion of TMSAB significantly improving prediction accuracy. Moreover, this generative approach, as opposed to direct prediction, not only adapts well to forecasting tasks but also provides a fresh perspective for other tasks in power systems.

REFERENCES

- J. W. Taylor and P. E. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 2213–2219, 2007.
- [2] J. Xie, Y. Chen, T. Hong, and T. D. Laing, "Relative humidity for load forecasting models," *IEEE Transactions on Smart Grid*, vol. 9, no. 1, pp. 191–198, 2016.
- [3] Y. Wang, Q. Xia, and C. Kang, "Secondary forecasting based on deviation analysis for short-term load forecasting," *IEEE Transactions* on *Power Systems*, vol. 26, no. 2, pp. 500–507, 2010.
- [4] H. Hou, C. Liu, Q. Wang, B. Zhao, L. Zhang, X. Wu, and C. Xie, "Load forecasting combining phase space reconstruction and stacking ensemble learning," *IEEE Transactions on Industry Applications*, vol. 59, no. 2, pp. 2296–2304, 2022.
- [5] S. Wang, X. Wang, S. Wang, and D. Wang, "Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting," *International Journal of Electrical Power & Energy Systems*, vol. 109, pp. 470–479, 2019.
- [6] L. Zhang, L.-y. Zhang, W. Jun et al., "Prediction of rolling load in hot strip mill by innovations feedback neural networks," *Journal of iron and* steel research, international, vol. 14, no. 2, pp. 42–51, 2007.
- [7] R. Ramanathan, R. Engle, C. W. Granger, F. Vahid-Araghi, and C. Brace, "Short-run forecasts of electricity loads and peaks," *International journal of forecasting*, vol. 13, no. 2, pp. 161–174, 1997.
- [8] J.-W. Xiao, P. Liu, H. Fang, X.-K. Liu, and Y.-W. Wang, "Short-term residential load forecasting with baseline-refinement profiles and biattention mechanism," *IEEE Transactions on Smart Grid*, 2023.
- [9] R. Koenker and K. F. Hallock, "Quantile regression," *Journal of economic perspectives*, vol. 15, no. 4, pp. 143–156, 2001.
- [10] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.
- [11] Y. Wang, D. Gan, M. Sun, N. Zhang, Z. Lu, and C. Kang, "Probabilistic individual load forecasting using pinball loss guided lstm," *Applied Energy*, vol. 235, pp. 10–20, 2019.
- [12] Z. Cao, C. Wan, Z. Zhang, F. Li, and Y. Song, "Hybrid ensemble deep learning for deterministic and probabilistic low-voltage load forecasting," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 1881– 1897, 2019.
- [13] X. Liu, L. Yang, and Z. Zhang, "The attention-assisted ordinary differential equation networks for short-term probabilistic wind power predictions," *Applied Energy*, vol. 324, p. 119794, 2022.
- [14] M. Sun, T. Zhang, Y. Wang, G. Strbac, and C. Kang, "Using bayesian deep learning to capture uncertainty for residential net load forecasting," *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 188–201, 2019.
- [15] B. Jiang, H. Yang, Y. Wang, Y. Liu, H. Geng, H. Zeng, and J. Ding, "Dynamic temporal dependency model for multiple steps ahead shortterm load forecasting of power system," *IEEE Transactions on Industry Applications*, 2024.
- [16] K. Qu, G. Si, Z. Shan, Q. Wang, X. Liu, and C. Yang, "Forwardformer: Efficient transformer with multi-scale forward self-attention for dayahead load forecasting," *IEEE transactions on power systems*, vol. 39, no. 1, pp. 1421–1433, 2023.
- [17] 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.
- [18] Y. Huang, H. Guo, E. Tian, and H. Chen, "Day-ahead probabilistic load forecasting: A multi-information fusion and noncrossing quantiles method," *IEEE Transactions on Industrial Informatics*, 2024.
- [19] A. Brusaferri, M. Matteucci, S. Spinelli, and A. Vitali, "Probabilistic electric load forecasting through bayesian mixture density networks," *Applied Energy*, vol. 309, p. 118341, 2022.
- [20] S. Haben and G. Giasemidis, "A hybrid model of kernel density estimation and quantile regression for gefcom2014 probabilistic load forecasting," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1017–1022, 2016.
- [21] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He, "Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies," *Transportation research part C: emerg*ing technologies, vol. 105, pp. 297–322, 2019.
- [22] Z. Wang, Q. Wen, C. Zhang, L. Sun, and Y. Wang, "Diffload: Uncertainty quantification in electrical load forecasting with the diffusion model," *IEEE Transactions on Power Systems*, 2024.
- [23] C. Luo, "Understanding diffusion models: A unified perspective," arXiv preprint arXiv:2208.11970, 2022.

- [24] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," Advances in neural information processing systems, vol. 33, pp. 6840–6851, 2020.
- [25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [27] B. Zhang, I. Titov, and R. Sennrich, "Improving deep transformer with depth-scaled initialization and merged attention," arXiv preprint arXiv:1908.11365, 2019.
- [28] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," arXiv preprint arXiv:2004.05150, 2020.
- [29] G. D. Rosin and K. Radinsky, "Temporal attention for language models," arXiv preprint arXiv:2202.02093, 2022.
- [30] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural networks*, vol. 107, pp. 3–11, 2018.
- [31] H. Wen, Y. Lin, Y. Xia, H. Wan, Q. Wen, R. Zimmermann, and Y. Liang, "Diffstg: Probabilistic spatio-temporal graph forecasting with denoising diffusion models," in *Proceedings of the 31st ACM International Confer*ence on Advances in Geographic Information Systems, 2023, pp. 1–12.
- [32] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.
- [33] 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.
- [34] 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.
- [35] P. Moreno, P. Ho, and N. Vasconcelos, "A kullback-leibler divergence based kernel for svm classification in multimedia applications," Advances in neural information processing systems, vol. 16, 2003.