Technical Appendix

An Efficient and Explainable Transformer-Based Few-Shot Learning for Modeling Electricity Consumption Profiles Across Thousands of Domains

August 19, 2024

Contents

In the following sections, we provide a detailed introduction to our model, present additional experiments and ablation studies, and discuss the performance of our method.

1 Loss Design

In this section, we detail the motivation behind our loss design. A naive approach to designing the loss function would involve directly applying the EM algorithm to the complete domain data to estimate the parameter θ_r. However, this approach has two major shortcomings 1) the θ_r estimate obtained through the EM algorithm is not globally optimal due to the inherent limitations of the EM algorithm, and 2) implementing the EM algorithm for every sample in each iteration significantly slows down the training process. Therefore, we utilize the negative log-likelihood as our loss function to bypass these issues, which is given by

$$\log \mathcal{L}(\mathcal{D}^k \mid \hat{\theta}_r, \mathbf{w}) = -\sum_{i=1}^{N} \log \left(\sum_{j=1}^{J} w_j \mathcal{N}(\mathbf{x}_i^k \mid \mu_j, \sigma_j) \right), \tag{1}$$

where $w_j \in \mathbf{w}$ is the fixed weight of the j-th Gaussian component, N is the number of data points in the domain, J is the number of Gaussian components, $\mathcal{N}(\mathbf{x}_i^k \mid \mu_j, \sigma_j)$ is the Gaussian probability density function for the j-th component with μ_j (mean vector) and σ_j.

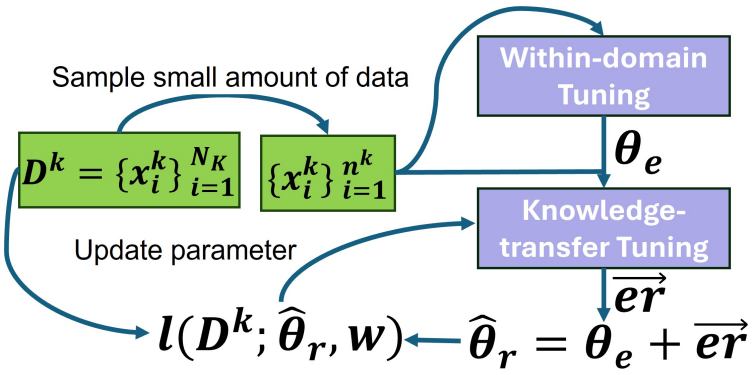


Figure 1: Training/inference process of one domain. $\hat{\theta}_r = \{\mu_j, \sigma_j\}_{j=1}^{J}$ represents the predicted parameter of GMMs, $\mathbf{w}$ is the weights of components, $l(\cdot)$ is the loss function. In the training process, $\mathcal{D}^k \in \mathcal{S}$. In the inference process, $\mathcal{D}^k \in \mathcal{T}$, and only $\hat{\theta}_r$ is predicted without loss computation and parameter updating for the Transformer.

2 Training and inference Algorithms

In this Section, we detailed the training and inference process which are summarized in Fig 1. In the main paper, we propose the following equation to determine z in Within-domain Tuning

$$z = \text{int}\left(e^{\beta n^k}\right), \tag{2}$$

where n^k is the number of ECP samples used in Within-domain Tuning, β is a parameter set to 0.015, which is the result of empirical testing in $\mathcal{S}$, int$(\cdot)$ means the integer part of a value. It is important to note that equation (2) serves as an empirical guideline for determining z in our specific experimental context. We advise practitioners to evaluate the optimal z for their datasets. In cases where further information is unavailable, we recommend setting z to either 1 or 2. With loss function in (1) and equation in (2), we can train the model using the algorithm shown in Algorithm 1. Similarly, for inference, we can use the algorithm shown in Algorithm 2.

Algorithm 1 Training

Require: Source domain collection $\mathcal{S}$, initial parameters θ_o, fixed weights $\mathbf{w}$, batch size L_s, the highest and lowest learning rates $l_h and l_w$.

Repeat
  Initialize parameters of GMMs $\theta = \theta_o$
  Sample L_s source domains from $\mathcal{S}$ to form the dataset $\{\mathcal{D}^k\}_{k=1}^{L_s}$
  From each source domain in $\{\mathcal{D}^k\}_{k=1}^{L_s}$, randomly sample n^{k_s} ECP samples, where $4 \leq n^{k_s} \leq 25$
  Compute the corresponding z using (2) for each sampled source domain
  Perform Within-domain Tuning to obtain θ_e for each source domain
  Perform Knowledge-transfer Tuning to obtain $\vec{er}$ for each source domain
  Compute the predicted $\hat{\theta}_r$ for each source domain
  Compute the loss in (1) and update the model parameters
Until convergence

Algorithm 2 Inference of One Target Domain

Require: One target domain $\mathcal{D}^{\|} = \{\mathbf{x_i}^k\}_{i=1}^{N}$, initial parameters θ_o, fixed weights $\mathbf{w}$, pre-trained Transformer encoder.

  Initialize parameters of GMMs $\theta = \theta_o$
  Compute the corresponding z using (2) and ECP samples in $\mathcal{D}$
  Perform Within-domain Tuning to obtain θ_e for the domain
  Perform Knowledge-transfer Tuning to obtain $\vec{er}$ for the domain
  Compute the predicted $\hat{\theta}_r$ for the domain

3 Implementation Details of Our Method

3.1 LogSumExp Trick

It is highly likely to encounter numerical stability issues during training using the loss function (1) and solving EM in batch (Within-domain Tuning). Therefore, we implement the LogSumExp trick to prevent instability. The loss function (1) has format like

$$\text{LogSumExp}(x_1, x_2, \ldots, x_n) = \log\left(\sum_{i=1}^{n} \exp(x_i)\right). \tag{3}$$

However, for numerical stability, we implement it as

$$\text{LogSumExp}(x_1, x_2, \ldots, x_n) = m + \log\left(\sum_{i=1}^{n} \exp(x_i - m)\right), \tag{4}$$

where $m = \max(x_1, x_2, \ldots, x_n)$. By using this trick, we ensure that our computations remain stable, even when dealing with extreme values, thereby improving the robustness of our training process.

3.2 Determining z with Varying n^k

In Section 2 and the main paper, we discussed how to determine the value of z using equation (2). However, when there are varying n^k values during training, the process of determining z becomes less straightforward (in contrast to when n^k is fixed). For our experiments in main paper, we used n^k values within the range $4 \leq n^k \leq 25$. In such cases, to minimize the risk of overfitting to the target domain, we recommend using the smallest possible value of n^k. For example, if $4 \leq n^k \leq 25$, selecting $n^k = 4$ would yield $z = 1$ according to equation (2).

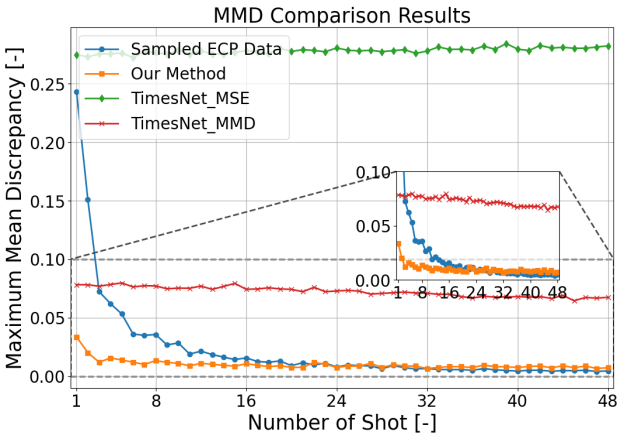


Figure 2: Comparison of MMD values on 30-minute resolution ECP dataset. We compute the average MMD values of n^{k_t} (number of shots) from 1 to 48.

3.3 Customized Sigmoid Function

Due to the σ_i (related to the covariance matrix) must be strictly positive, and because we scale ECP samples to [0,1] during training, the output μ_i, σ_i should both be positive. To ensure this, we apply a Sigmoid function at the end of the output layers, expressed as

$$\text{CustomSigmoid}(x) = a \times \left(\frac{1}{1 + e^{-\alpha(x-\beta)}} \right) + b \tag{5}$$

where α is the scaling factor, β is the shifting factor, a scales the output, and b shifts the output. In our case, we set $\alpha = 1.0$, $\beta = 0.5$, $a = 1$, and $b = 0.000001$. You can check the provided repository with the path `gmm_trans_package/model/gmm_transformer.py` for detailed code realization.

Table 1: Results of evaluation metrics in the validation set.

Method	MMD	KL	KS	WD	MSE.M
4-shot					
Sampled ECP	0.1246	1.2658	0.4040	0.1460	0.0030
TimesNet MSE	0.2995	1.4999	0.3120	0.2191	0.0130
TimesNet MMD	0.0896	0.5281	0.2448	0.0972	0.0055
Our Method	0.0607	0.4922	0.2800	0.1214	0.0015
8-shot					
Sampled ECP	0.0570	0.8109	0.3987	0.1318	0.0014
TimesNet MSE	0.2992	1.5017	0.3080	0.2125	0.0131
TimesNet MMD	0.0890	0.4949	0.2279	0.0926	0.0052
Our Method	0.0367	0.3540	0.1776	0.0746	0.0007
16-shot					
Sampled ECP	0.0324	0.6685	0.3036	0.1004	0.0008
TimesNet MSE	0.3003	1.4285	0.3088	0.2103	0.0131
TimesNet MMD	0.0817	0.4887	0.2312	0.0943	0.0048
Our Method	0.0317	0.3957	0.1880	0.0779	0.0008
32-shot					
Sampled ECP	0.0238	0.4811	0.2000	0.0721	0.0008
TimesNet MSE	0.3021	1.4789	0.3112	0.2118	0.0132
TimesNet MMD	0.0757	0.5588	0.2352	0.0908	0.0046
Our Method	0.0284	0.3545	0.2000	0.0747	0.0006

3.4 Other Details

The proposed method is order-agnostic, meaning that swapping the order of input ECP samples (or ECP token order) does not affect the output. Therefore, operations such as batch normalization should not be included in the model, as they would compromise this characteristic.

4 Implementation of TimesNet

In the main paper, we present the evaluation of TimesNet using two distinct loss functions. We adopt the time series imputation model from the original paper in `https://github.com/thuml/Time-Series-Library.git`. However, the imputation task of TimesNet is originally trained with MSE loss which is defined as

$$\mathcal{L}_{\text{MSE}} = \sum_{i=1}^{L} (y_i - \hat{y}_i)^2, y_i \in \mathbb{R}^{N^k \times T}, \hat{y}_i \in \mathbb{R}^{N^k \times T}, \tag{6}$$

where L is the batch size, y_i is the i-th original time series data and $\hat{y}_i$ is the i-th imputed time series data. N^k is the number of ECP samples in the k-th domain. Our primary objective in this paper is to model the distribution of ECP data rather than to accurately impute the energy consumption time series. The MSE loss implicitly takes into account the order of ECP samples, which may impose unnecessary constraints on the model. To mitigate this issue, we employ the MMD as the loss function to train the model. MMD is defined as

$$\begin{aligned} \mathcal{L}^2_{\text{MMD}} =& \frac{1}{N^k(N^k-1)} \sum_i \sum_{j \neq i} k(y_i, y_j) - \frac{2}{N^k N^k} \sum_i \sum_j k(y_i, \hat{y}_j) \\ &+ \frac{1}{N^k(N^k-1)} \sum_i \sum_{j \neq i} k(\hat{y}_i, \hat{y}_j), \end{aligned} \tag{7}$$

where $k(\cdot)$ is the kernel function applied to pairs of data points. In this paper, we adopt the Radial Basis Function kernel, which is commonly used

$$k(\hat{y}_i, y_i) = \exp\left(-\frac{\|\hat{y}_i - y_i\|^2}{2\sigma^2}\right), \tag{8}$$

where σ is a hyper-parameter which we set to 1 in our experiments.

5 Additional Experiments

5.1 Experiment Setting

5.1.1 Data

While the main paper utilizes hourly resolution ECP data, this section focuses on 30-minute resolution data, which exhibits greater volatility. This dataset comprises data from approximately 20 thousand households in the UK and Australia. We use the same data augmentation method described in the main paper to have a dataset of around 40 thousand domains (each domain has 250 ECP samples). We carefully split the data into training, testing, and validation sets with a ratio of 0.8, 0.1, and 0.1, ensuring that the augmented domain remains within the same set. Similar to the main dataset, this dataset will also be made available as open-source.

5.1.2 Experiment Design

Similar to the main paper, we use TimesNet MMD and TimesNet MSE as a baseline to evaluate our method's performance. In the training training process, we set $8 \leq n^{k_s} \leq 48$ (approximate 3.2% to 20% percentage of complete dataset), for inference we set $1 \leq n^{k_t} \leq 48$.

5.2 Experiments Results

In this section, we present additional experimental results. Fig 5 presents a portion of the experimental results, illustrating that our method more effectively reproduces the peak, valley values, and overall pattern of ECP data compared to TimesNet. Fig 2 presents the MMD comparison results for n^{k_t} (number of shots) ranging from 1 to 48. The results indicate that TimesNet struggles to accurately model the ECP data distribution. Additionally, we observe that when $n^{k_t} > 26$, the sampled ECP data distribution closely aligns with the original ECP distribution, surpassing our method. This finding may suggest a boundary to the effectiveness of our approach. Table 1 presents a quantitative summary of the comparison results, demonstrating that the predicted GMM outperforms other alternatives in the 32-shot scenario, achieving the best performance in KS, MSE, and KL metrics. Additionally, it surpasses all other methods in the 4, 8, and 16-shot scenarios.

6 Ablation Study

6.1 Experiment Setting

In this section, we provide a detailed evaluation of the impact of Within-domain Tuning and Knowledge-transfer Tuning. To assess the effectiveness of Within-domain Tuning, we conducted experiments where this component was omitted, and the Transformer encoder alone was used to directly predict $\vec{or}$, followed by the calculation of $\hat{\theta}_r = \vec{or} + \theta_o$. For Knowledge-transfer Tuning, we mentioned the design choice of removing the encoder's positional encoding to enhance learning efficiency, as discussed in the main paper. To evaluate this design, we train a model employing the classical absolute positional encoding from [1] in the encoder. All models used in the ablation experiments were trained and evaluated at a similar scale. To conduct this study efficiently, we trained tiny models with approximately 35 thousand parameters, and surprisingly, our method remained effective even at this reduced scale.

6.2 Within-domain Tuning

Fig 4a and 4c illustrate the MMD values for n^{k_t}, ranging from 1 to 24 and 1 to 48, respectively, comparing the performance of our method with and without Within-domain Tuning at hourly and 30-minute resolutions. The results demonstrate that the model's performance deteriorates without Within-domain Tuning. This is likely due to the large distance $||\theta_o - \theta_r||$ in the parameter space of GMMs, as discussed in the main paper, which can negatively impact the model's accuracy. Furthermore, Fig 4b and 4d show the results of the 8-shot experiment.

6.3 Knowledge-transfer Tuning

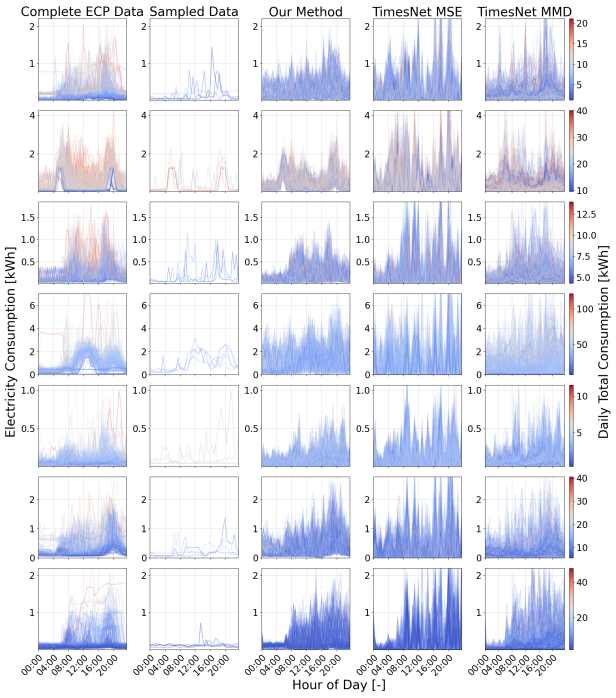
Similarly, Fig 5a and 5c present the MMD values for different numbers of shots, comparing the performance of our method with and without positional encoding in the encoder. The MMD comparison clearly shows that positional encoding affects the model's performance. As discussed in the main paper, each ECP sample is treated as an individual token, and for GMMs, each sample is supposed to be independent and identical. Introducing positional encoding essentially introduces noise into the model, thereby impacting its performance. Fig 5b and 5b illustrate the results of the 8-shot experiment.

7 Deeper Discussion of Our Method

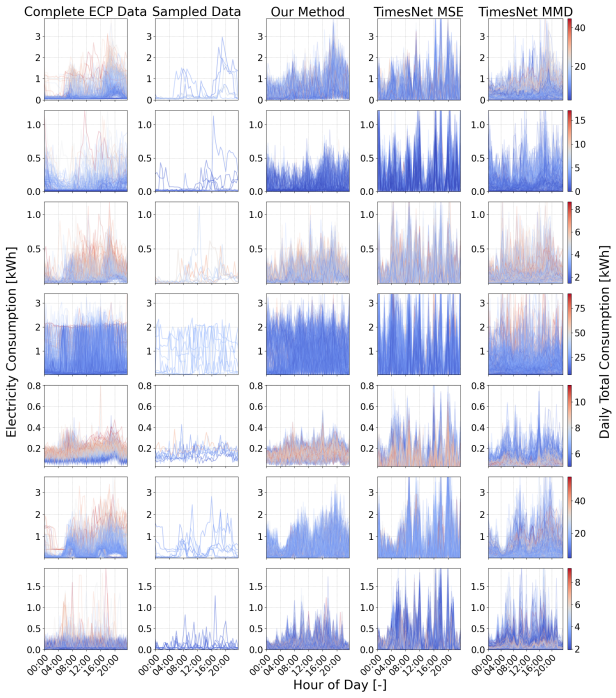
In this section, we aim to give a deeper explanation of our method by answering some questions that we might not have answered in the main paper.

Question 1: Why learning to predict GMMs rather than directly learning an end-to-end DL model generates a better performance?

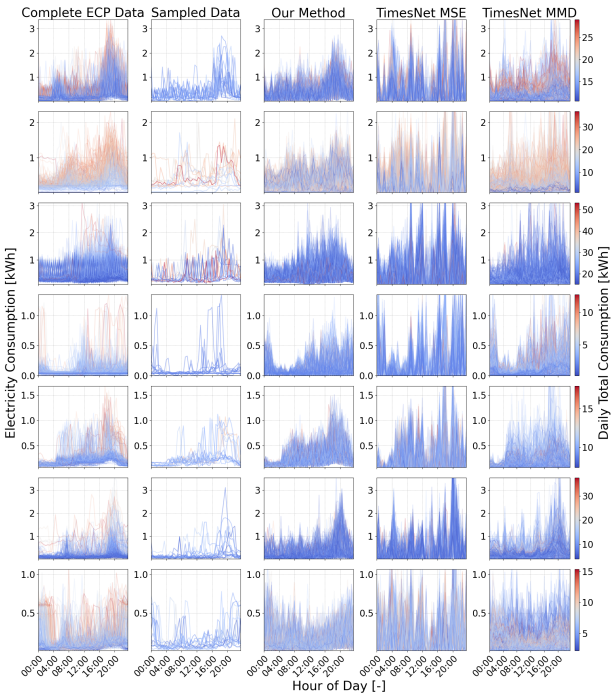
We think GMMs can be essentially regarded as an effective distribution encoder and decoder in our experiments. GMMs encode the ECP distribution using a set of parameters, μ and σ, and the process of decoding this distribution involves sampling from the GMMs defined by these parameters. Consequently,



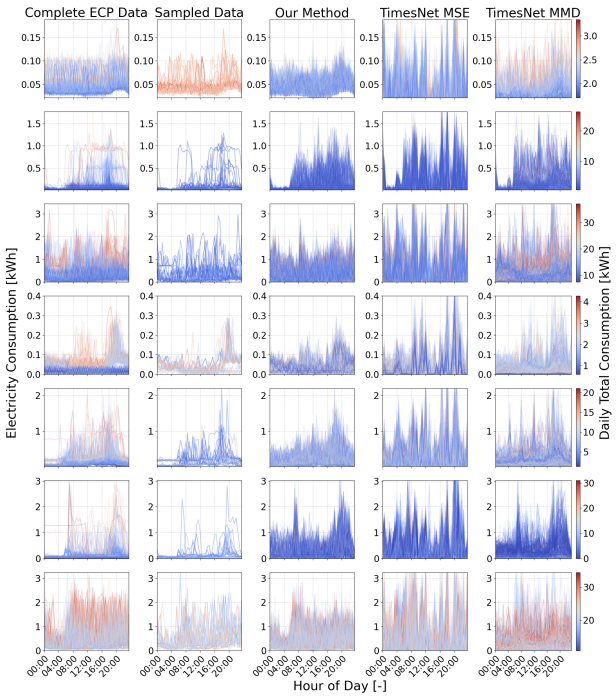
(a) 4-shot ($n^{k_t} = 4$).



(b) 8-shot ($n^{k_t} = 8$).

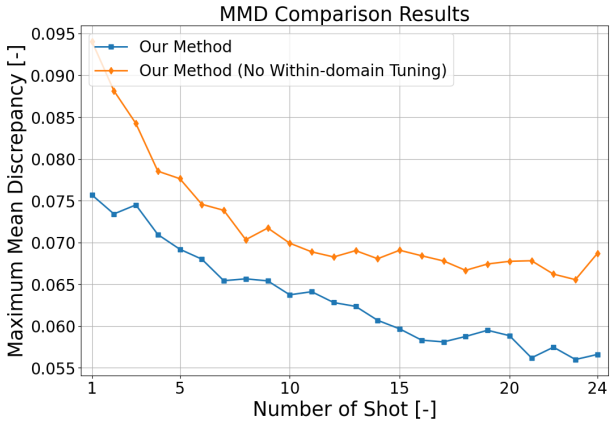


(c) 16-shot ($n^{k_t} = 16$).

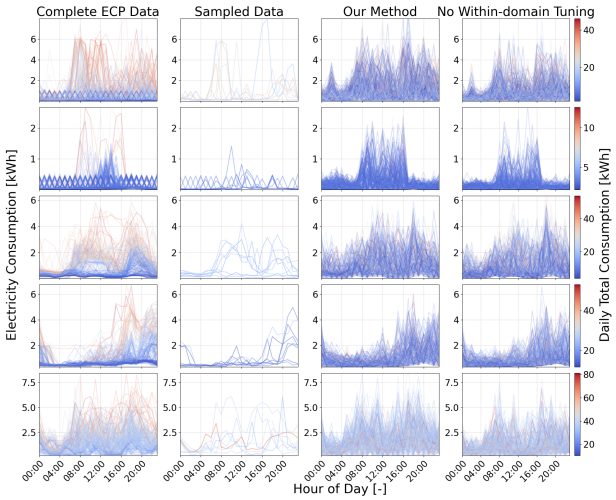


(d) 32-shot ($n^{k_t} = 32$).

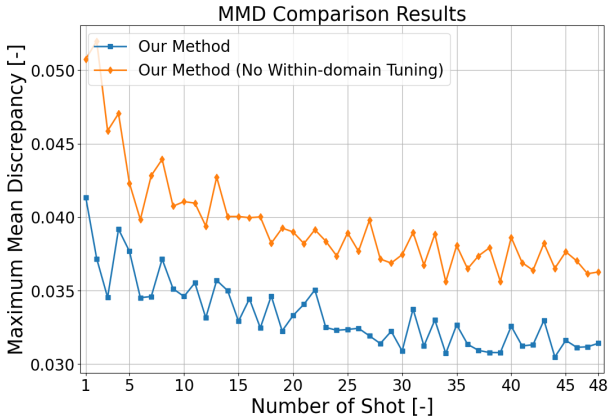
Figure 3: In each subfigure, every row represents the experimental results for a specific target domain. Each row, from left to right, includes (1) the complete ECP data of the domain, (2) the sampled ECP data used as input for our model and two TimesNets, (3) the ECP data generated from GMMs whose parameters are predicted by our method, (4) results generated by TimesNet using MSE loss, and (5) results generated by TimesNet using MMD loss. The color of the curves is related to the daily total electricity consumption.



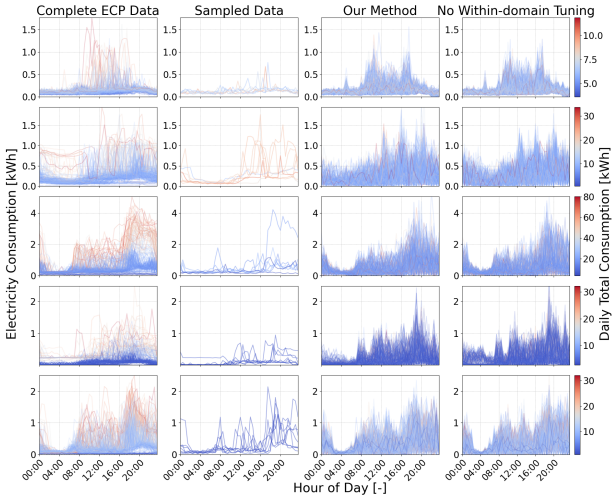
(a) Comparison of with and without Within-domain Tuning (hourly resolution).



(b) 8-shot ECP modeling result of hourly resolution data ($n^{k_t} = 8$).

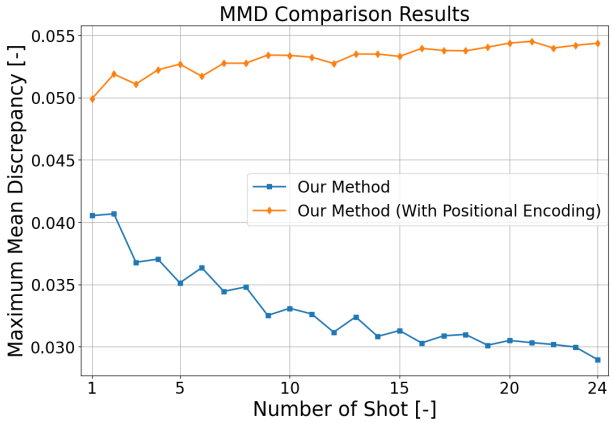


(c) Comparison of with and without Within-domain Tuning (30-minutes resolution).

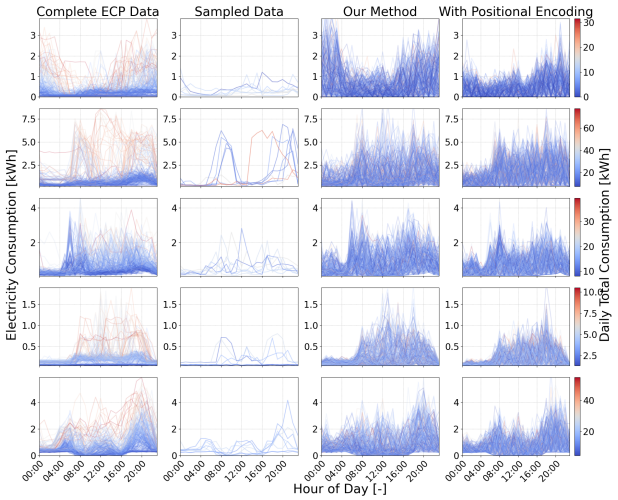


(d) 8-shot ECP modeling result of 30-resolution resolution data ($n^{k_t} = 8$).

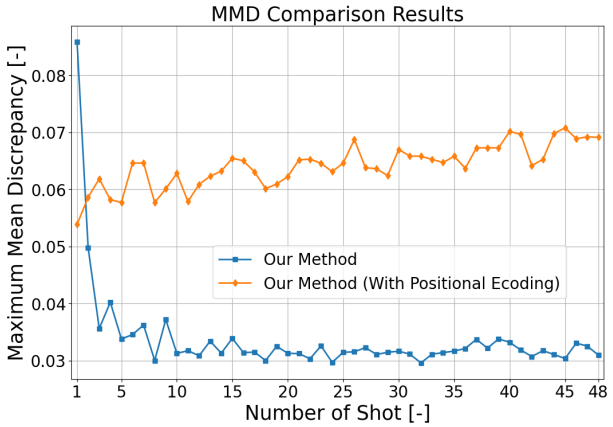
Figure 4: Subfigure (a) and (c) show MMD comparison results. In subfigure (b) and (d), every row represents the experimental results for a specific target domain. Each row, from left to right, includes (1) the complete ECP data of the domain, (2) the sampled ECP data used as input for our model and two TimesNets, (3) the ECP data generated from GMMs whose parameters are predicted by our method with Within-domain Tuning, and (4) the ECP data generated from GMMs whose parameters are predicted by our method without Within-domain Tuning. The color of the curves is related to the daily total electricity consumption.



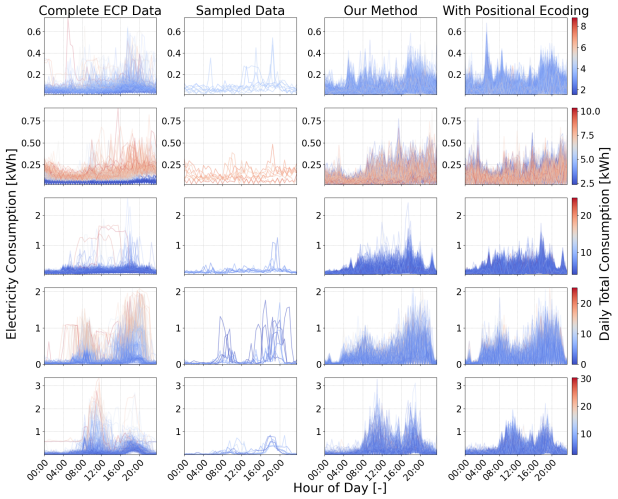
(a) Comparison of without and with positional encoding (hourly resolution).



(b) 8-shot ECP modeling result of hourly resolution data ($n^{k_t} = 8$).



(c) Comparison of without and with positional encoding (30-minutes resolution).



(d) 8-shot ECP modeling result of 30-minutes resolution data ($n^{k_t} = 8$).

Figure 5: Subfigure (a) and (c) show MMD comparison results. In subfigure (b) and (d), every row represents the experimental results for a specific target domain. Each row, from left to right, includes (1) the complete ECP data of the domain, (2) the sampled ECP data used as input for our model and two TimesNets, (3) the ECP data generated from our method without positional encoding in the encoder, and (4) the ECP data generated from our method with positional encoding in the encoder. The color of the curves is related to the daily total electricity consumption.

compared to directly recovering the ECP dataset based on some input ECP samples (as done by TimesNet in our experiments), learning the parameters of the GMMs can be seen as learning a latent representation, which may be more efficient.

Question 2: Why use a Transformer to predict the parameter of GMMs?

We think the transformer has some inherent advantages for predicting the parameters of GMMs. In the main paper, as well as in the ablation studies, we have demonstrated that a Transformer encoder without positional encodings can serve as an efficient architecture for this task from the perspective of ***iid*** data. Moreover, the attention mechanism within the Transformer computes a weighted sum of the token values, which can be seen as analogous to how means and variances are derived for a distribution. Therefore, as we designed in the main paper, μ, σ, and ECP samples are considered as tokens, allowing the Transformer encoder to effectively capture potential mean and variance changes given a limited set of input ECP samples.

Question 3: What is the limitation of our method?

The limitations of our method are closely tied to the inherent limitations of GMMs. While GMMs are effective at modeling the mean and variance of distributions, they struggle with capturing the temporal correlations in time series data (or other types of data). Additionally, the assumption of Gaussian distributions may hinder the model's ability to accurately represent more complex, non-Gaussian distributions.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.