iEnergy

ARTICLE

https://doi.org/10.23919/IEN.2022.0041

Adaptive probabilistic load forecasting for individual buildings

Chenxi Wang[1], Dalin Qin[1], Qingsong Wen[2], Tian Zhou[3], Liang Sun[2] and Yi Wang[1] ✉

ABSTRACT

Building-level load forecasting has become essential with the support of fine-grained data collected by widely deployed smart meters. It acts as a basis for arranging distributed energy resources, implementing demand response, etc. Compared to aggregated-level load, the electric load of an individual building is more stochastic and thus spawns many probabilistic forecasting methods. Many of them resort to artificial neural networks (ANN) to build forecasting models. However, a well-designed forecasting model for one building may not be suitable for others, and manually designing and tuning optimal forecasting models for various buildings are tedious and time-consuming. This paper proposes an adaptive probabilistic load forecasting model to automatically generate high-performance NN structures for different buildings and produce quantile forecasts for future loads. Specifically, we cascade the long short term memory (LSTM) layer with the adjusted Differential ArchiTecture Search (DARTS) cell and use the pinball loss function to guide the model during the improved model fitting process. A case study on an open dataset shows that our proposed model has superior performance and adaptivity over the state-of-the-art static neural network model. Besides, the improved fitting process of DARTS is proved to be more time-efficient than the original one.

KEYWORDS

Probabilistic load forecasting, long short-term memory (LSTM), Differentiable neural ARchiTecture Search (DARTS), building load forecasting.

Electric load forecasting is of great importance for power system planning and operation. Accurate load forecasts support system operators to better or even optimally schedule generators to supply the demand and thus reduce the overall operational costs of the systems[1]. In recent years, the widely deployed smart meters record fine-grained electricity consumption data both temporally and spatially, which makes building-level load forecasting possible. Predicting loads for individual buildings helps to track the dynamic change in demand and thus acts as basis for arranging distributed energy resources[2], implementing demand response[3], etc. Therefore, building-level load forecasting is as crucial for flexibility provision and renewable energy accommodation.

However, building-level load forecasting is faced with two major challenges: (1) Compared with aggregated-level load, the electric load of individual buildings tends to be more uncertain and fluctuating[4]. Traditional point forecasts fail to demonstrate such uncertainty. (2) Since buildings of different usage see distinct electricity consumption patterns, as shown in Figure 1, a well-designed forecasting model for one building may be unsuitable for others. In addition, manually designing and tuning optimal forecasting models for various buildings can be tedious and time-consuming.

Therefore, we are motivated to develop an effective and efficient method to cope with these two challenges. To handle the first challenge, we generate quantiles of future loads by quantile regression to give information about uncertainty; while for the second one, we utilize the state-of-the-art Differentiable neural ARchiTecture Search (DARTS)[5] technique to adaptively adjust forecasting models for buildings of variety.

In the past decades, building-level load forecasting has received much attention. Statistical and machine learning techniques, such as ARIMA[6], support vector regression (SVR)[3], artificial neural networks (ANN)[7,8], are frequently applied. Among these techniques, ANN is becoming more and more welcomed due to the rising deep neural networks, which have more powerful non-linear fitting capability and flexible network structures. ResNet, a model of convolutional neural networks (CNN), is used for feature fusion when generating load forecasts[9]. Besides, recurrent neural networks (RNN) model is applied for household load forecasting[10]. Through the test on household data from Ireland, RNN is proved to have superior performance to ARIMA and SVR. Long short-term memory (LSTM)-based sequence to sequence model is also used for learning the long and short-term patterns of historical load data and generating a prediction of arbitrary steps[11]. Mogrifier LSTM[12] is applied to improve the performance of LSTM in load forecasting in refs. [13] and [14]. Combining CNN and LSTM, a hybrid CNN–LSTM model in ref. [15] performs feature extraction through CNN and then carries out sequence learning by LSTM, showing superiority over standalone LSTM.

In recent years, the research focus has shifted from point load forecasting to probabilistic load forecasting. Conventional deterministic load forecasting only outputs the expected value of the loads in the future, while probabilistic load forecasting produces intervals, quantiles, or density of the future loads to provide more information about future uncertainties[16]. To generate probabilistic forecasts, three major methods can be concluded[16], i.e., (1) generating multiple input scenarios by simulation[17,18], (2) utilizing probabilistic mathematics models (such as quantile regression)[19], and (3) advancing point forecasts into probabilistic ones via residual simulation or forecast combination[20]. In the context of building load forecasting, the latter two approaches are predominantly used. Modeled by the Gaussian process, both deterministic and probabilistic forecasts are produced for residential loads in ref. [21].

[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China; [2]DAMO Academy, Alibaba Group (U.S.) Inc., Bellevue, WA 98004, USA; [3]DAMO Academy, Alibaba Group, Hangzhou 310023, China
Address correspondence to Yi Wang, yiwang@eee.hku.hk

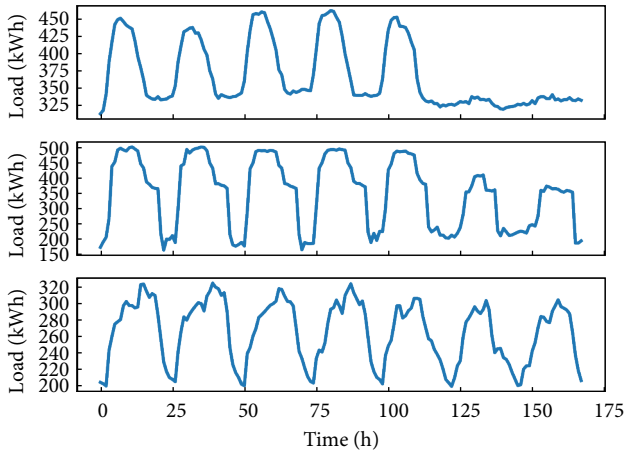


Fig. 1 Different electricity consumption patterns of different buildings. (up: one education building, mid: one public building, down: one lodging building)

Combining with quantile regression, a pinball-loss-guided LSTM model is constructed for individual load forecasting[22]. Through implementing dropout techniques as a variational Bayesian approximation[23], builds Bayesian RNN to forecast loads of an educational building in Hong Kong. Consisting of probabilistic normal load forecasting and the probabilistic peak abnormal differential load forecasting, the final forecasts are presented for buildings in ref. [24].

Apparently, with the help of ANN, there is an increasing number of studies in building-level load forecasting either in the deterministic or probabilistic forms. These existing studies, however, mainly focus on designing and tuning their models for one particular customer or class of customers, which may lead to (1) cumbersome and time-consuming developing process for a machine learning model and (2) a forecasting model that lacks flexibility, robustness and adaptivity.

Thanks to the emerging neural architecture searching (NAS) techniques, such problems can be alleviated. NAS becomes a hot topic in automated machine learning (AutoML). Since connections and operations (e.g. activation function, convolution, etc.) significantly affect the performance of a neural network, NAS focuses on searching them for neural networks in the given search space, resulting in a robust and high-performance neural architecture[25]. There are three popular optimization methods used in NAS, i.e., reinforcement learning[26,27], evolutionary algorithm[28], and gradient descent[5,29]. The former two both sample the architecture in a discrete search space for searching, while the latter one directly searches in a continuous and differentiable space, significantly reducing the search time. DARTS, a representative method of gradient descent-based NAS, relaxes the discrete search space into a continuous and differentiable one through the softmax function[5]. DARTS and subsequential algorithms have been widely used for image classification and language processing. However, in the context of load forecasting, their potential has not yet been further explored. In ref. [30], the researchers naively generate RNN cells through original DARTS, and merely obtains competitive results with the RNN family in the task of short-term load forecasting. Therefore, we would like to further explore the probabilistic load forecasting for individual buildings of variety with the support of NAS techniques, to provide an effective and efficient model with adaptivity and thus enrich literature offerings in both energy and AI fields.

The main contributions of this paper are as follows:

- Propose a novel and adaptive probabilistic load forecasting model for individual buildings by cascading a classic LSTM layer and a DARTS module, which can automatically generate high-performance NN structures for customers with diverse consumption patterns.
- Improve the original DARTS model by adjusting the training process from the perspective of fine-tuning and including an early-stopping mechanism in both the searching and training processes, which reduces computation time significantly and enhances the practicality.
- Conduct a comprehensive case study based on open datasets including 15 buildings of 5 distinct usages, which verifies and analyzes the superiority, adaptivity, and time efficiency of the proposed model compared to state-of-the-art pinball loss guided LSTM (QLSTM) and the original DARTS (ODARTS).

The rest of the paper is organized as follows. The proposed adaptive probabilistic forecasting model is introduced in Section 1. The benchmark and evaluation metrics for the forecasts are presented in Section 2. Case studies and numerical analysis are shown in Section 3. Finally, conclusions are drawn in Section 4.

1 Adaptive probabilistic load forecasting

The proposed model for adaptive probabilistic load forecasting will be introduced in this section. Overall model structure, detailed model components, and complete model fitting process will be included.

1.1 Model structure

To realize the goal of adaptive probabilistic load forecasting, our proposed model consists of three core components, i.e., the LSTM layer, the DARTS cell, and the pinball loss function. The former two support building the adaptive forecasting networks while the last one helps to generate the probabilistic forecasts. The overall model structure is demonstrated in Figure 2, and the details of each component will be discussed in the subsequent subsections.

The first component is the LSTM layer, which is a prestigious neural network for handling time-series data. An LSTM layer consists of LSTM units that share the same determined structure and are arranged in a sequence. Thus, the first component of our model is a neural network layer with fixed and static architecture, responsible for receiving the sequential data at different timestamps as inputs, learning the pattern of inputs timestep by timestep, and outputting the extracted features in the last time step for the following module.

The second component is the DARTS cell. Unlike the fixed structure in the LSTM layer, the DARTS cell is initially made up of neural nodes that have not yet decided on their connections and operations with each other, as the dashed line shown in Figure 2. This cell will search in the designed search space containing the alternative operations. Then, it will select the optimal ones for the nodes through gradient descent, and thus form the most suitable neural network for each individual building as the red line, as shown in Figure 2. Consequently, this cell realizes the adaptive learning for various buildings.

The third component is the pinball loss function. To produce probabilistic forecasts, we adopt the form of quantiles. The pinball loss function will guide the built network throughout the model fitting process to update the trainable weights. After the fitting process, the final model will be able to generate quantiles of the future loads and thus form the final probabilistic forecasts for buildings.

With these components, our model is able to learn the temporal dependency of loads data, generate the adaptive NN blocks and make quantile load forecasts for individual buildings.

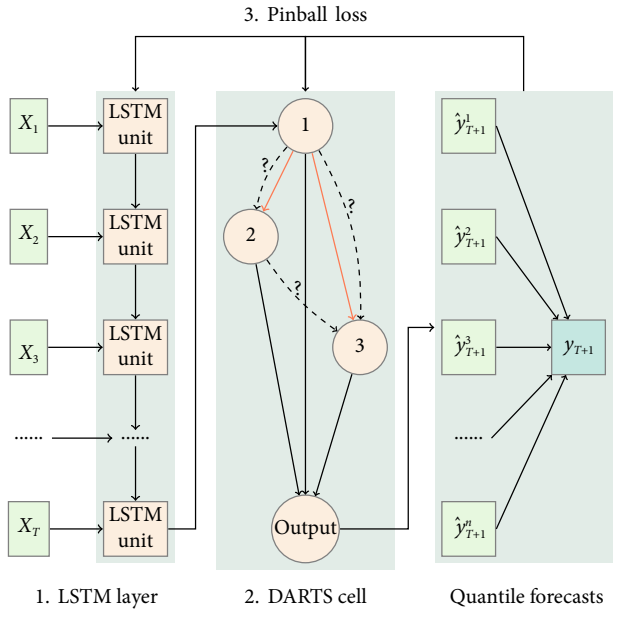


Fig. 2 Overall model structure.

1.2 LSTM

LSTM is a classic model in the RNN family, with the ability to memory both short-term and long-term information from time series data. Vanilla RNN uses the hidden states h to learn information from the time series. However, the hidden states are prone to the latest input, leading to the failure to remember the long-term information. In LSTM, in addition to the hidden states h, there is the cell state c to help capture long-term temporal information, as shown in Figure 3(a). Thus, LSTM is able to keep both short-term and long-term information with the support of both h and c.

Looking into the LSTM unit shown in Figure 3(b), there are three gates to control the information flow, i.e. forget gate, input gate, and output gate. realizing long short-term memory. The first gate is the forget gate f_t, controlling how much information will be erased from the previous cell state c_t. The forget gate is computed as follows:

$$f_t = \sigma((X_t, h_{t-1}) \cdot W_f), \tag{1}$$

where σ denotes the sigmoid function; X_t is the input data at time t, including historical load, air temperature, and calendar features; h_{t-1} represents the previous hidden state; W_f stands for the weights matrix in the forget gate to be trained. After calculation, f_t is result vector for the forget gate at time t.

The second gate is the input gate, which controls how much information will be added into the next cell state c_t from h_{t-1} and X_t. Similarly, through the sigmoid function, the input gate vector can be computed as:

$$i_t = \sigma((X_t, h_{t-1}) \cdot W_i). \tag{2}$$

Constricted by these two gates, the cell state will be updated as follows:

$$g_t = \tanh((X_t, h_{t-1}) \cdot W_g), \tag{3}$$

$$c_t = i_t \odot g_t + f_t \odot c_{t-1} \tag{4}$$

where tanh denotes tanh activation function; $\odot$ means the element-wise product; g_t is the calculation cache that is used for the updates of the cell state.

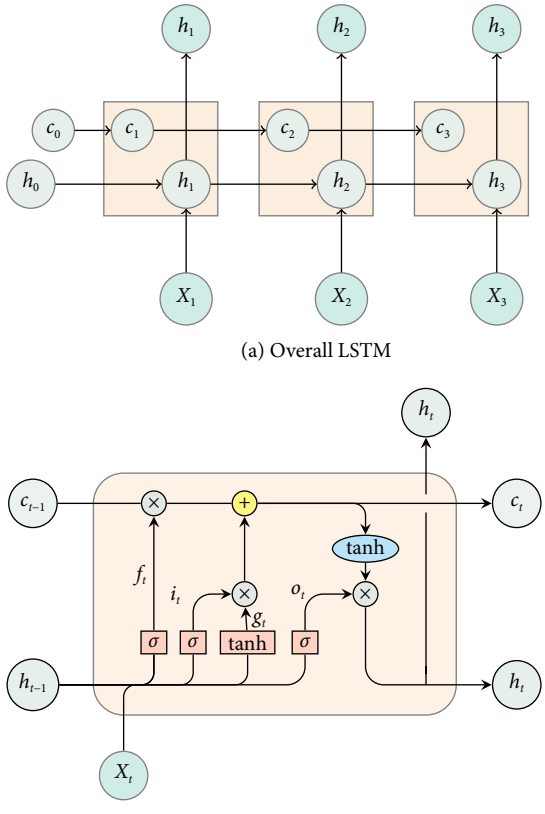


Fig. 3 LSTM structure.

The last gate is the output gate, controlling how much information will be presented as final outputs, and the final output is the hidden state at time t, h_t, dependant on the updated cell state c_t. Thus, the final output of the LSTM unit at time t is formulated as:

$$o_t = \sigma((X_t, h_{t-1}) \cdot W_o), \tag{5}$$

$$h_t = \tanh(o_t \odot c_t). \tag{6}$$

1.3 Adjusted DARTS

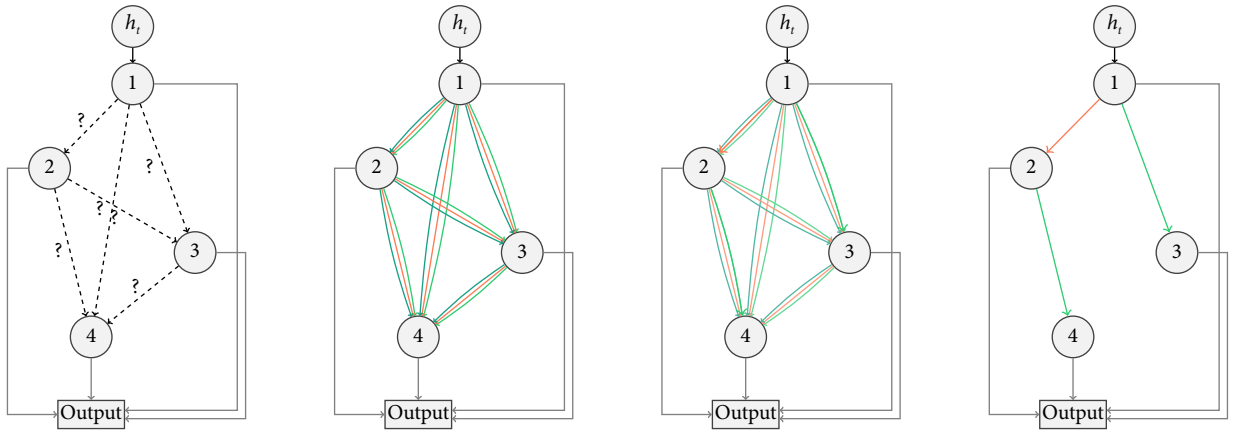
The DARTS cell is used to automatically generate high-performance NN structures for individual buildings. In our model, it takes the latest output of the LSTM layer as input and then outputs the quantile forecasts finally. A DARTS cell is composed of nodes and edges. Each node $x^{(i)}$ represents the latent tensors, such as feature maps in CNN, while each edge $o^{(i,j)}$ connects node i and j, representing the operation used to transform $x^{(i)}$ to $x^{(j)}$. Meanwhile, for each node, it is connected with all the previous nodes and computed as:

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)}). \tag{7}$$

For the last node in the cell, it averages all the outputs of previous nodes and then outputs:

$$x^{(n)} = \frac{1}{n-1} \sum_{i=1}^{n-1} x^{(i)}. \tag{8}$$

Initially, the operations in the edges are unknown as illustrated in Figure 4(a) and there are some alternatives to be selected, constructing the search space. In our work, we simplify the DARTS cell for the conventional ANN so the search space is the activation



(a) Operations are unknown initially (b) Relax the search space through mixing operations on each edge (c) Jointly optimize the architecture and weights in the network (d) Derive the final architecture from the learned architecture parameters

Fig. 4 Overall DARTS.

functions. In addition, the none operation, i.e., two nodes are not connected, is included in the search space as well. Therefore, deciding the operations on the edges also implies the connection relationship of nodes. Consequently, the goal of DARTS is to decide which operation will be selected on each edge.

To fulfill this goal, DARTS relaxes its discrete search space to a continuous one by parameterizing the operations and applying the softmax function to all possible operations on each edge (as shown in Figure 4(b)). The specific method is shown below:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'}^{(i,j)})} o(x), \tag{9}$$

where $\bar{o}^{(i,j)}(x)$ is the mixed operation of all candidates; O is the set of all possible operations; $\alpha_o^{(i,j)}$ is the parameter representing the importance of the operation $o \in O$ on the edge from node i to node j. In this way, the searching task is reduced to learning the set of continuous α. After searching, the most likely operation will be selected for each edge, and then the final discrete architecture will be derived by retaining the strongest operation for each node, as indicated in Figure 4(d). The strength of an operation is defined as $\frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'}^{(i,j)})}$, i.e., the coefficient of each operation in Eq. (9).

To jointly learn the architecture α and weights w of the model, a bilevel optimization problem is proposed as follows in ref. [5]:

$$\begin{aligned} \min_{\alpha} \quad & L_{\text{val}}(w^*(\alpha), \alpha) \\ s.t. \quad & w^*(\alpha) = \operatorname{argmin}_w L_{\text{train}}(w, \alpha) \end{aligned} \tag{10}$$

To deal with such optimization problem, Algorithm 1 is also designed in ref. [5] to alternating α and w updates on the validation and test sets respectively.

Algorithm 1 DARTS

Require: learning rate η_α and η_w for α and w, loss function L, learning rate for stepping ahead ξ

1: Initialize the DARTS cell.

2: **while** not converged **do**

3: $\alpha - = \eta_\alpha \nabla_\alpha L_{\text{val}}(w - \xi \nabla_w L_{\text{train}}(w, \alpha), \alpha)$ ($\xi = 0$ if using the first-order approximation)

4: $w - = \eta_w \nabla_w L_{\text{train}}(w, \alpha)$

5: **end while**

Ensure: Discrete architecture based on α

In the original work of DARTS, the none operation is excluded when deriving the architecture, for (1) fairly comparing their DARTS with other state-of-the-art NAS algorithms and (2) avoiding the situation of all none operations when the searching process finishes. However, excluding the none operation means losing the connection relationship among the nodes. Thus, in our work, we made two following adjustments:

(1) We include the none operation when obtaining our NN structures to retain all the possibility of architecture searching since we do not focus on the comparison of NAS algorithms.

(2) We assign the operation from the LSTM output to the first node in the DARTS cell as the conventional sigmoid activation, ensuring basic fitting ability even if the searching results are no operation.

In this way, our model not only guarantees a bottom line of fitting capability but also gives full flexibility and adaptability through DARTS.

1.4 Loss function

Since we aim to produce the quantiles of future loads, the pinball loss function is used to guide the model fitting instead of the mean squared error loss function. The pinball loss function is computed as follows:

$$L_\tau(\hat{y}_t^\tau, y_t) = \begin{cases} (1-\tau)(\hat{y}_t^\tau - y_t) & y_t < \hat{y}_t \\ \tau(y_t - \hat{y}_t^\tau) & y_t \geq \hat{y}_t \end{cases} \tag{11}$$

where τ denotes the target quantile; $\hat{y}_t^\tau$ denotes the predicted q-th quantile of the load at time t; y_t denotes the true load at time t.

Obviously, the original pinball loss function is not differentiable at the zero point and thus hinders us from using gradient descent optimization methods. Thus, we apply Huber norm[31] to the pinball loss function to make it differentiable everywhere. The Huber norm can be calculated as follows:

$$h(\hat{y}_t, y_t) = \begin{cases} \frac{(\hat{y}_t - y_t)^2}{2\varepsilon}, & 0 \leq |\hat{y}_t - y_t| \leq \varepsilon \\ |\hat{y}_t - y_t| - \frac{\varepsilon}{2}, & |\hat{y}_t - y_t| > \varepsilon \end{cases} \tag{12}$$

where ε is a threshold. When the prediction error is less than ε, the Huber norm will be presented as the L2 norm, and vice versa, it will be the L1 norm. In this way, the approximate pinball loss function is differentiable at the zero point, as shown below:

$$L_\tau'(\hat{y}_t^\tau, y_t) = \begin{cases} (1-\tau)\, h(\hat{y}_t^\tau, y_t) & y_t < \hat{y}_t \\ \tau h(\hat{y}_t^\tau, y_t) & y_t \geq \hat{y}_t \end{cases} \tag{13}$$

To produce relatively complete quantile forecasts, we usually need to provide a number of quantiles, such as at $\tau = 0.1, 0.2, \cdots, 0.9$. However, fitting the model for each quantile adds a much computational burden. Thus, we output multiple quantiles at the same time, calculate the average pinball loss of all quantiles and adjust the loss function to the following composite form:

$$L(\hat{y}_t, y_t) = \frac{1}{n_\tau} \sum_{\tau=1}^{n_\tau} L_\tau'(\hat{y}_t^\tau, y_t) \tag{14}$$

In this way, our proposed model can be fitted only once to produce multiple quantiles. Compared with fitting n_τ times for an individual quantile model, this adjustment reduces the computational time to $\frac{1}{n_\tau}$ times of the original one.

1.5 Improved three-staged model fitting

The complete model fitting process in our model is three-staged: searching, selecting, and training. Before the model fitting process, the test dataset will be split and used only at the end of the third stage to report the final performance of the model for the individual building. The whole process is illustrated in Figure 5.

In the searching stage, we first initialize n models with different random seeds. Then, the DARTS searching algorithm will be conducted for each model M_i to search its architecture. We add an early stop mechanism to the searching stage, which monitors the descent of validation loss and stops searching in advance when the loss does not drop for a given threshold for the number of epochs. Thus, Algorithm 1 can be improved as Algorithm 2. With the early stop mechanism, the searching process will avoid unnecessary computation and alleviate the overfitting.

Algorithm 2 DARTS training with early stop

Require: learning rate η_α and η_w, loss function L, learning rate for stepping ahead ξ, maximum searching epoch N, patience epoch N_p

1: **while** not reach N epoch **do**

2: $\alpha - = \eta_\alpha \nabla_\alpha L_{\text{val}}(w - \xi \nabla_w L_{\text{train}}(w, \alpha), \alpha)$ ($\xi = 0$ if using the first-order approximation)

3: $w - = \eta_w \nabla_w L_{\text{train}}(w, \alpha)$

4: **if** L_{val} dosen't decrease for N_p epoch **then**

5: break

6: **end if**

7: **end while**

Ensure: Discrete architecture based on α

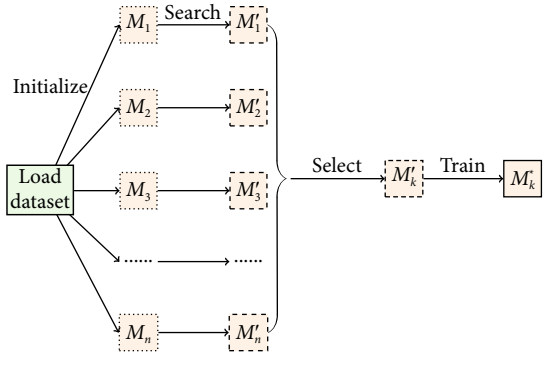


Fig. 5 Overall fitting process.

After searching, we obtain n discrete DARTS cells and respective forecasting models. It has to be mentioned that since the DARTS algorithm is designed to jointly optimize both the α and w in the searching stage, randomly initializing the weights w after searching is unnecessary in ref. [5]. Thus, we keep the learned weights in these models. Then, we train these models from scratch on the training dataset for a short period and report their performance on the validation dataset. Based on their performance, we pick the best-searched model M_k' for the final training. In this way, we can avoid the initialize-sensitive optimization results of the architecture searching and acquire a relatively robust model for training.

The final stage is training the model. We also keep the learned weights of the picked model in the second stage and continue to train this model M_k' in the perspective of fine-tuning. Similar to the searching stage, we add an early stop mechanism in this stage as well. After the training, we obtain the completely fitted model M_k^* and report the model performance on the test dataset which is not been used before at all.

2 Evaluation metrics and benchmarks

In this section, we introduce the metrics to evaluate the final performance of the probabilistic forecasts and the benchmarks to compare with our proposed model.

2.1 Evaluation metrics

To evaluate the probabilistic forecasts, two comprehensive metrics, quantile score and Winkler score, are introduced.

The quantile score (QS), is defined as the mean of the pinball loss across the whole predicted series. Thus, it can be calculated as

$$QS = \frac{1}{n_\tau \cdot T} \sum_{t=1}^{T} \sum_{\tau=1}^{n_\tau} L_\tau(\hat{y}_t, y_t) \tag{15}$$

A lower score indicates a better probabilistic forecast.

The Winkler score (WS) is a metric for assessing the prediction intervals (PI). For a central (1-α)% PI, it is defined as follows:

$$WS_{\alpha,t} = \begin{cases} \delta, & L_t \leq y_t \leq U_t \\ \delta + \dfrac{2(y_t - U_t)}{\alpha}, & y_t > U_t \\ \delta + \dfrac{2(L_t - y_t)}{\alpha}, & y_t < L_t \end{cases} \tag{16}$$

where, L_t and U_t represent the lower and upper bound of the PI respectively; $\delta = U_t - L_t$. The WS actually penalizes when the PI doesn't cover the observation value and rewards for the narrow PI. Similarly, a lower WS means a better PI.

2.2 Benchmarks

In this paper, we compare our proposed model with QLSTM, a powerful neural network used to forecast the quantiles of the target variables. Compared with our model, there is no adaptive DARTS cell for the outputs in QLSTM. If DARTS searching results in an all-none cell, i.e. all nodes in the cell are suspended and the LSTM output directly pass through the DARTS cell, our model will degrade to QLSTM as shown in Figure 6.

Besides, we also include the ODARTS, the original DARTS, as one of the benchmarks. It is designed to have the same setups except for the fitting process. Compared with our proposed model, ODARTS re-initializes the weights after the searching and selecting stage and does not adopt the early stop mechanism in the whole model fitting process.

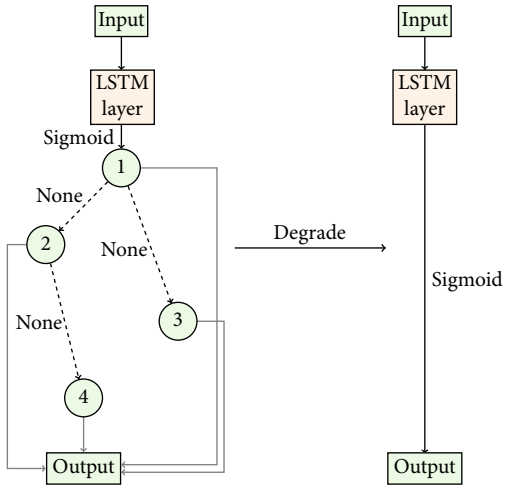


Fig. 6 Degradation of DARTS.

3 Case study

In the case study, we focus on 24h-ahead probabilistic load forecasting. The probabilistic forecasts, adaptive model architectures and computation time comparison will be analyzed in the subsections.

3.1 Experimental setups

The dataset used to construct our case study is BDG2 dataset[32], including the energy data from 1636 buildings. The sampling frequency is hourly measurements of electricity. The time range of the dataset is two complete years (2016 and 2017).

Among the 1636 buildings, we filter the buildings whose electricity recordings are not complete. Then, we randomly select 15 buildings of 5 different usages as the final customers in the case study, as illustrated in Table 1. The last 744 data points (the last month) in the datasets are held for testing the models while the first 16800 data points are used for model fitting.

We implement all the forecasting models and benchmarks through Tensorflow. To achieve parallel computing of multiple models, we apply Ray, a package of distributed computing, in the model fitting stage of the proposed model. The parallel accelerating happens in the model searching and selecting stage as demonstrated in *Improved three-staged model fitting* in Section 1. All the experiments are supported by one Inter(R) Xeon(R) W-3335 CPU @ 3.40GHz.

The settings of our proposed model, i.e., mixing LSTM with DARTS (denoted as MDARTS), and the competing benchmarks are listed in Table 2. The setups of MDARTS and ODARTS are the same except for the fitting process while QLSTM lacks a DARTS cell compared to MDARTS. For DARTS-related models, we initialize $n = 5$ models with different random seeds as mentioned in Section 1.5.

All models undergo the same feature engineering with inputs of historical load, temperature and calendar variables (month, day, weekday, hour) for the last 7 days. They are also set to produce $n_\tau = 9$ quantiles of future building loads, from 0.1 to 0.9, forming the final probabilistic forecasts.

Table 1 The selected datasets in the USA

Usage	Dataset
Education	Rachael
	Ricardo
	Donnie
Industry	Jeremy
	Mariah
	Joanne
Lodging	Ora
	Shanti
	Hal
Office	Valda
	Shawnna
	Bill
Public	Crystal
	Gerard
	Kevin

3.2 Probabilistic forecasts

Table 3 and Table 4 respectively, show the comparison on QS and WS (for PI = 80%) of the proposed MDARTS and benchmarks on the 15 datasets. The best performances on each dataset are highlighted in bold and the average performances of each model on all data sets are highlighted in italics.

Compared with QLSTM, MDARTS performs significantly better regarding QS over the 15 various datasets and achieves an improvement of 6.33% on average. Similarly, on WS, despite the slight performance lag on Education usage datasets, MDARTS still made a 5.51% improvement. Figure 7 presents a scatter plot of the detailed pinball loss on all quantiles from 0.1 to 0.9 on all the datasets between MDARTS and QLSTM. Apparently, most of the points lie under the $y = x$ line, which indicates that MDARTS outperforms the QLSTM on almost all quantiles. To demonstrate the actual forecasting results, Figure 8 presents the probabilistic forecast results for one week (from December 11th to December 18th in 2017) on dataset Gerard. The black dots indicate the observed building loads, and the blue-filled areas imply the probabilistic forecasts. It is obvious that the prediction intervals produced by MDARTS appear to be narrow and basically capture the dynamic changes of building loads, while the ones generated by QLSTM are relatively wide and loose and fail to track the load trends on some data points. Similar results can also be found on other datasets. Thus, compared with QLSTM, the proposed MDARTS shows its superiority in probabilistic performance.

Compared to ODARTS, MDARTS outperforms in the average

Table 2 Important parameters

Model	Architecture parameters				Fitting parameters				
	Search space	DARTS nodes	LSTM layer	Hidden units	Search epoch	Small train epoch	Final train epoch	Optimizer	Batchsize
QLSTM	NA				NA				
ODARTS	{none, sigmoid, tanh, relu, gelu, linear}	3	1	16	60 (reinitialize)	50 (reinitialize)	300	adam	32
MDARTS					60 (early stop)	50 (early stop)			

Table 3 Quantile score comparison

Usage	Dataset	Quantile score			Improvement	
		MDARTS	QLSTM	ODARTS	vs QLSTM	vs ODARTS
Education	Rachael	**2.061**	2.124	2.136	3.06%	3.64%
	Ricardo	**5.370**	5.487	5.394	2.18%	0.45%
	Donnie	**1.99**	2.052	2.01	3.12%	1.01%
Industry	Jeremy	**1.731**	1.877	1.802	8.43%	4.10%
	Mariah	**0.297**	0.318	0.298	7.07%	0.34%
	Joanne	**5.237**	5.390	5.431	2.92%	3.70%
Lodging	Ora	**4.350**	4.642	4.540	6.71%	4.37%
	Shanti	**8.809**	9.712	9.092	10.25%	3.21%
	Hal	**2.352**	2.556	2.698	8.67%	14.71%
Office	Valda	**1.027**	1.09	1.052	6.13%	2.43%
	Shawnna	**12.644**	12.927	13.885	2.24%	9.81%
	Bill	5.632	6.206	**5.212**	10.19%	−7.46%
Public	Crystal	**5.021**	5.472	5.574	8.98%	11.01%
	Gerard	1.265	1.398	**1.214**	10.51%	−4.03%
	Kevin	**11.698**	12.625	13.119	7.92%	12.15%
AVG		*4.632*	*4.925*	*4.897*	*6.33%*	*5.72%*

Table 4 Winkler score comparison

Usage	Dataset	Winkler score (PI = 80%)			Improvement	
		MDARTS	QLSTM	ODARTS	vs QLSTM	vs ODARTS
Education	Rachael	26.641	26.427	**26.439**	−0.80%	−0.76%
	Ricardo	72.280	70.393	**65.827**	−2.61%	−8.93%
	Donnie	24.591	**23.801**	25.735	−3.21%	4.65%
Industry	Jeremy	**20.784**	23.467	22.033	12.91%	6.01%
	Mariah	**3.434**	3.693	3.917	7.54%	14.07%
	Joanne	**64.143**	72.558	75.507	13.12%	17.72%
Lodging	Ora	**54.166**	59.07	56.365	9.05%	4.06%
	Shanti	116.665	126.329	**113.997**	8.28%	−2.29%
	Hal	**29.339**	31.835	34.680	8.51%	18.20%
Office	Valda	**13.616**	14.989	14.41	10.08%	5.83%
	Shawnna	**159.486**	168.600	175.252	5.71%	9.89%
	Bill	80.708	83.382	**71.763**	3.31%	−11.08%
Public	Crystal	**67.976**	76.114	83.873	11.97%	23.39%
	Gerard	16.009	19.075	**14.615**	19.15%	−8.71%
	Kevin	**171.142**	172.036	189.121	0.52%	10.51%
AVG		*61.399*	*64.785*	*64.902*	*5.51%*	*5.71%*

forecasting performance, despite its poor performance on several datasets in terms of WS and fails to beat ODARTS on Bill and Gerard regarding QS. Over the 15 datasets, MDARTS improves by at least 5.71% in both QS and WS compared to ODARTS.

To further analyze the performance of our proposed MDARTS for each building, we make a scatter plot as shown in Figure 9. The usage of buildings is also colored distinctly for analysis. It can be seen that there is no remarkable relationship in either the quantile score and the usages and the improvements and the usages. QS varies from building to building because of the differences in their level of electricity consumption. Even buildings with the same usage may have significant differences in electricity consumption. Similarly, though MDARTS improves the forecasting performance of the test buildings, there is no strong correlation between the improvements and the building usages. In some type of buildings, both QS and improvements can be significantly varing, such as *Office* buildings shown in Figure 9.

3.3 Adaptive model architectures

With the support of DARTS cells, our proposed MDARTS generates different high-performance NN structures for the tested 15 datasets and Figure 10 shows all the structures learned by

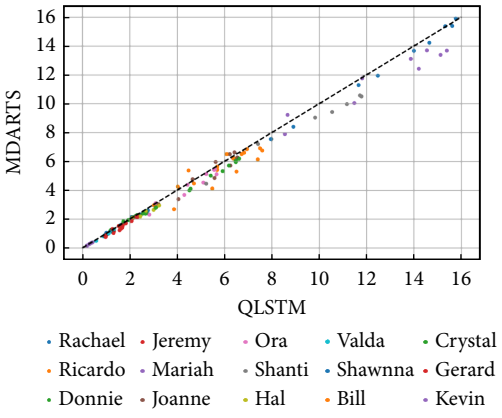


Fig. 7 Pinball loss comparison between MDARTS and QLSTM.

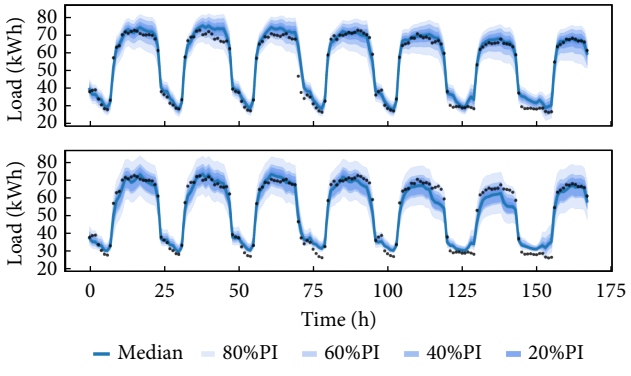


Fig. 8 Forecasts on Gerard. (up: MDARTS, down: QLSTM)

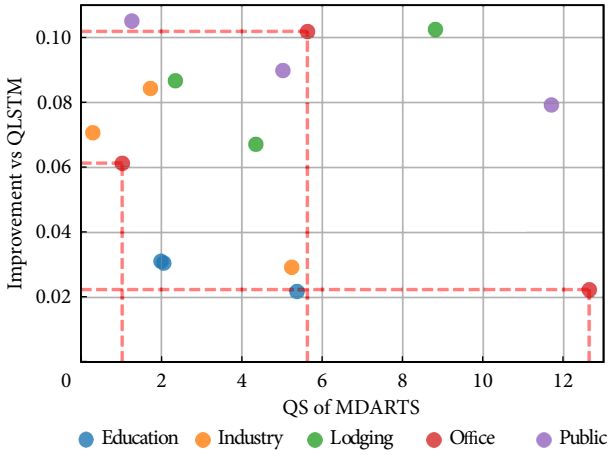


Fig. 9 Forecasting performance of our proposed MDARTS for each building.

MDARTS for each buildings. It has to be mentioned that since the structure of the LSTM layer in front of the DARTS cells is unified and stationary, we draw only the DARTS cells in Figure 10, and the inputs of the cells are the outputs of LSTM. Since the DARTS nodes are set as 3 in Table 2, the structures in the Figure 10 has up to three nodes.

It can be seen that each building consumer obtains their own model architecture automatically generated by MDARTS, which saves many manual designs. Theoretically, there are 15 structures for 15 buildings but several customers share the same structures, such as Shawnna and Hal, Mariah and Ora. Thus, there are 9 different model structures in the end. It is also interesting that there are five building customers, i.e. Donnie, Rachael, Jeremy, Valda, and Gerard, obtaining the degraded architecture, which is the most searched structure and the same as QLSTM. However, they still enjoy significant improvements in probabilistic forecasting compared to using QLSTM. The possible reason for this phenomenon is that the joint optimization of the search phase provides relatively better initial weights than randomly initialized for the subsequent training of the discrete model. To illustrate this thought, Figure 11 presents the loss curves of these five degraded MDARTS and QLSTM, where the blue curve represents MDARTS, and the orange one represents QLSTM. Figure 11 shows that at the end of the search, there is a small abrupt change in the loss curve of MDARTS, which is caused by the discretization from the mixing structure. Then, compared to randomly initialized QLSTM, MDARTS tends to start with a relatively lower loss start point and continue its fitting process, which is owing to keeping the learned weights after the searching stage.

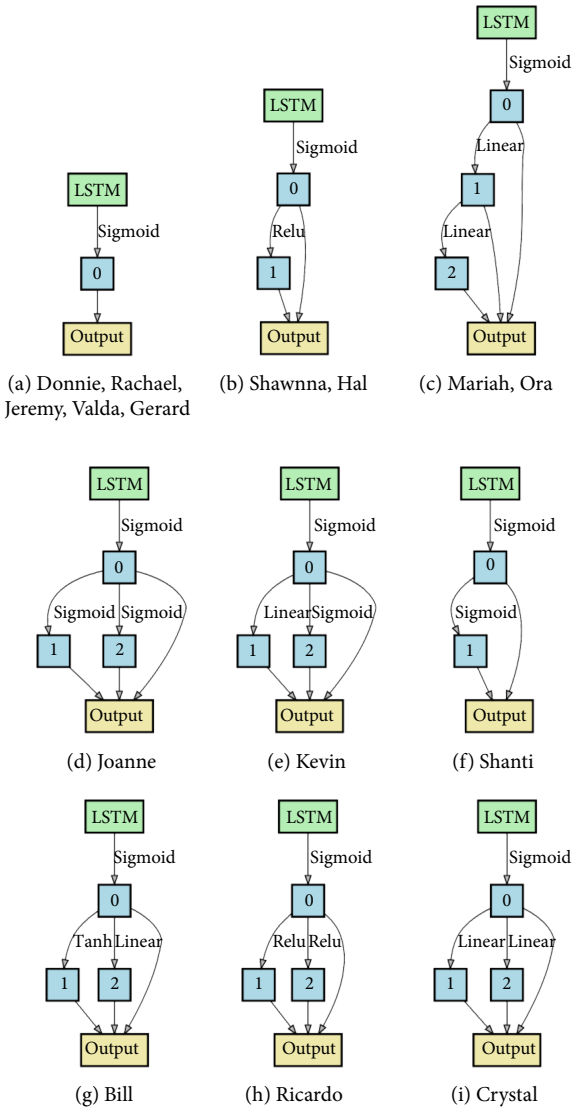


Fig. 10 DARTS cells generated for different buildings.

3.4 Computation time

Figure 12(a) shows the boxplot of three models over all datasets to illustrate the computation time for the model fitting process. All the models are implemented on the same hardware and software. It can be seen that although MDARTS is more time-consuming

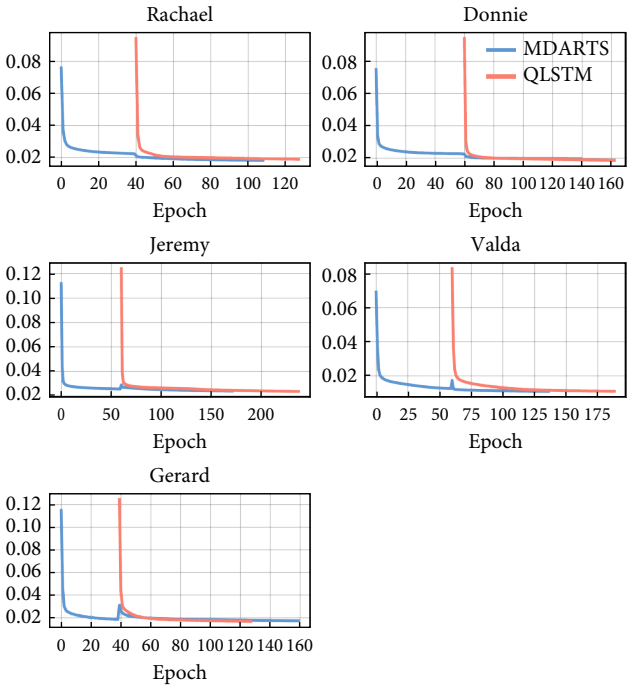
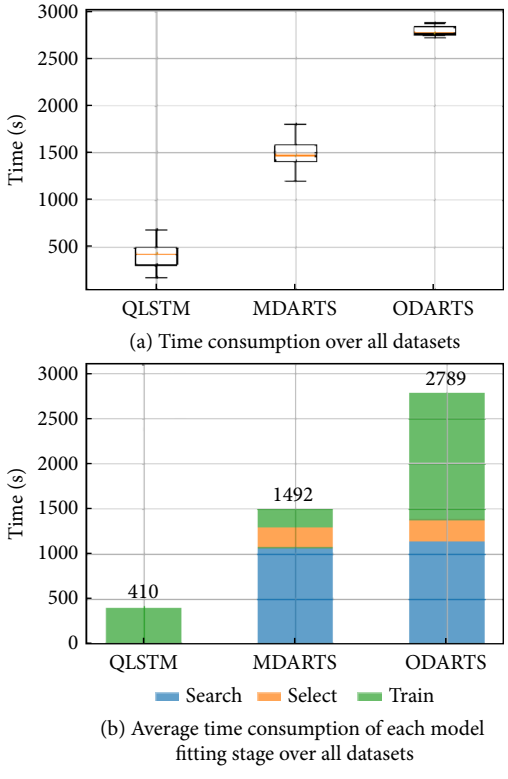


Fig. 11 Loss curves on degraded MDARTS and QLSTM.



(a) Time consumption over all datasets

(b) Average time consumption of each model fitting stage over all datasets

Fig. 12 Computation time comparison.

than QLSTM, it is much more time-efficient compared to ODARTS. Besides, MDARTS has a wider range of computation time than ODARTS, which is due to the different stop epochs of the early stop mechanism on different datasets.

To dig into the model fitting process, we compute the average elapsed time of each stage for the QLSTM, MDARTS, and ODARTS on all data sets. The results are shown in Figure 12(b), and the average total computation time is also annotated in the figure. Without the searching and selecting process, QLSTM takes 410 s, less than 7 min, to finish the whole fitting process, while DARTS-related models take at least three times more time to complete the three-stage fitting task. Compared to ODARTS, although MDARTS did not save significant time in the search and selection stage, it takes much less time in the final training stage. Overall, MDARTS saves 46.4% of computation time over ODARTS.

The whole-staged training loss curve is shown in Figure 13. In ODARTS, the network weights are reinitialized at the beginning of both the selecting and training stages. In contrast, the weights in MDARTS are trained throughout the model fitting process, and they are not forgotten during the whole process, thus shortening the number of epochs required for the final training. In this way, the long computation time problem of DARTS can be alleviated, and the practicality of DARTS is thus enhanced.

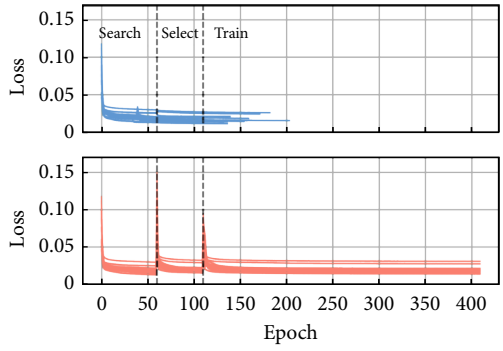


Fig. 13 Three-staged fitting process over all datasets. (up: MDARTS, down: ODARTS)

4 Conclusions and future works

In this paper, we proposed a novel adaptive probabilistic forecasting model, i.e., MDARTS, for individual buildings by incorporating the LSTM layer and DARTS cell. We also conduct a comprehensive case study based on an open dataset and compare our proposed model with the competing benchmarks (QLSTM and ODARTS) on the metrics of quantile score and Winkler score. We can draw a three-fold conclusion from the case study results:

(1) In terms of probabilistic forecasting, MDARTS outperforms both QLSTM and ODARTS on an average of 15 building datasets. Regarding the two metrics, the improvements on QLSTM and ODARTS are at least 5.51%, implying the superiority of MDARTS on probabilistic forecasting.

(2) MDARTS automatically produces high-performance neural network structures for each building. While some of the MDARTS degrade to QLSTM, the consumers still enjoy the forecasting improvement for the possible reason that the additional model searching stage gives MDARTS a better starting point for subsequent training.

(3) By keeping the weights learned in the search and select stage and adding the early stop mechanism, MDARTS needs fewer epochs than ODARTS during the model fitting process and thus saves 46.4% computation time, which enhances its practicality.

For future works, we will include two aspects: (1) increasing the number of the building datasets and the type of building usages to explore the relationship between the model structures and the building usages; (2) developing the study of scalability to further handle the large datasets and the extensive model searching spaces.

Acknowledgement

The work was supported in part by the Seed Fund for Basic Research for New Staff of The University of Hong Kong (202107185032) and in part by the Alibaba Innovative Research programme.

Article history

Received: 17 August 2022; Revised: 2 October 2022; Accepted: 8 October 2022

Additional information

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Hong, T. (2014). Energy forecasting: Past, present, and future. *Foresight: The International Journal of Applied Forecasting*, 32: 43–48.

[2] Mocanu, E., Nguyen, P. H., Gibescu, M., Kling, W. L. (2016). Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 6: 91–99.

[3] Chen, Y. B., Xu, P., Chu, Y. Y., Li, W. L., Wu, Y. T., Ni, L. Z., Bao, Y., Wang, K. (2017). Short-term electrical load forecasting using the support vector regression (SVR) model to calculate the demand response baseline for office buildings. *Applied Energy*, 195: 659–670.

[4] Yu, C. N., Mirowski, P., Ho, T. K. (2017). A sparse coding approach to household electricity demand forecasting in smart grids. *IEEE Transactions on Smart Grid*, 8: 738–748.

[5] Liu, H., Simonyan, K., Yang, Y. (2018). Darts: Differentiable architectureresearch. *arXiv preprint*, 1806.09055.

[6] Nepal, B., Yamaha, M., Yokoe, A., Yamaji, T. (2020). Electricity load forecasting using clustering and ARIMA model for energy management in buildings. *Japan Architectural Review*, 3: 62–76.

[7] Dagdougui, H., Bagheri, F., Le, H., Dessaint, L. (2019). Neural network model for short-term and very-short-term load forecasting in district buildings. *Energy and Buildings*, 203: 109408.

[8] Kim, Y., Son, H. G., Kim, S. (2019). Short term electricity load forecasting for institutional buildings. *Energy Reports*, 5: 1270–1280.

[9] Wang, J. S., Chen, X. H., Zhang, F., Chen, F. X., Xin, Y. (2021). Building load forecasting using deep neural network with efficient feature fusion. *Journal of Modern Power Systems and Clean Energy*, 9: 160–169.

[10] Shi, H., Xu, M. H., Li, R. (2018). Deep learning for household load forecasting —A novel pooling deep RNN. *IEEE Transactions on Smart Grid*, 9: 5271–5280.

[11] Marino, D. L., Amarasinghe, K., Manic, M. (2016). Building energy load forecasting using deep neural networks. In: Proceedings of the IECON 2016 —42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy.

[12] Melis, G., Kočiskỳ, T., Blunsom, P. (2019). Mogrifier LSTM. *arXiv preprint*, 1909.01792.

[13] Shen, X. D., Zhao, H. X., Xiang, Y., Lan, P., Liu, J. Y. (2022). Short-term electric vehicles charging load forecasting based on deep learning in low-quality data environments. *Electric Power Systems Research*, 212: 108247.

[14] Tan, B., Ma, X., Shi, Q. H., Guo, M., Zhao, H. X., Shen, X. D. (2021). Ultra-short-term wind power forecasting based on improved LSTM. In: Proceedings of the 2021 6th International Conference on Power and Renewable Energy (ICPRE), Shanghai, China.

[15] Alhussein, M., Aurangzeb, K., Haider, S. I. (2020). Hybrid CNN–LSTM model for short-term individual household load forecasting. *IEEE Access*, 8: 180544–180557.

[16] Hong, T., Fan, S. (2016). Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32: 914–938.

[17] Hong, T., Wilson, J., Xie, J. R. (2014). Long term probabilistic load forecasting and normalization with hourly information. *IEEE Transactions on Smart Grid*, 5: 456–462.

[18] Taylor, J. W., Buizza, R. (2002). Neural network load forecasting with weather ensemble predictions. *IEEE Transactions on Power Systems*, 17: 626–632.

[19] Zhang, W. J., Quan, H., Srinivasan, D. (2019). An improved quantile regression neural network for probabilistic load forecasting. *IEEE Transactions on Smart Grid*, 10: 4425–4434.

[20] Liu, B. D., Nowotarski, J., Hong, T., Weron, R. (2017). Probabilistic load forecasting via quantile regression averaging on sister forecasts. *IEEE Transactions on Smart Grid*, 8: 730–737.

[21] Shepero, M., van der Meer, D., Munkhammar, J., Widén, J. (2018). Residential probabilistic load forecasting: A method using Gaussian process designed for electric load data. *Applied Energy*, 218: 159–172.

[22] Wang, Y., Gan, D. H., Sun, M. Y., Zhang, N., Lu, Z. X., Kang, C. Q. (2019). Probabilistic individual load forecasting using pinball loss guided LSTM. *Applied Energy*, 235: 10–20.

[23] Xu, L., Hu, M. M., Fan, C. (2022). Probabilistic electrical load forecasting for buildings using Bayesian deep neural networks. *Journal of Building Engineering*, 46: 103853.

[24] Xu, L., Wang, S. W., Tang, R. (2019). Probabilistic load forecasting for buildings considering weather forecasting uncertainty and uncertain peak load. *Applied Energy*, 237: 180–195.

[25] He, X., Zhao, K. Y., Chu, X. W. (2021). AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212: 106622.

[26] Zoph, B., Le, Q. V. (2016). Neural architecture search with reinforcementlearning. *arXiv preprint*, 1611.01578.

[27] Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J. (2018). Efficient neural architecture search via parameters sharing. In: Proceedings of the 35th International Conference on Machine Learning.

[28] Real, E., Aggarwal, A., Huang, Y. P., Le, Q. V. (2019). Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33: 4780–4789.

[29] Shin, R., Packer, C., Song, D. X. (2018). DARTS: Differentiable neural network architecture search. Available at https://openreview.net/forum?id=S1eYHoC5FX.

[30] Biju, G. M., Pillai, G. N., Seshadrinath, J. (2019). Electric load demand forecasting with RNN cell generated by DARTS. In: Proceedings of the TENCON 2019—2019 IEEE Region 10 Conference, Kochi, India.

[31] Huber, P. J. (2009). Robust Statistics. Hoboken, NJ, USA: John Wiley & Sons.

[32] Miller, C., Kathirgamanathan, A., Picchetti, B., Arjunan, P., Park, J. Y., Nagy, Z., Raftery, P., Hobson, B. W., Shi, Z. X., Meggers, F. (2020). The Building Data Genome Project 2, energy meter data from the ASHRAE Great Energy Predictor III competition. *Scientific Data*, 7: 368.