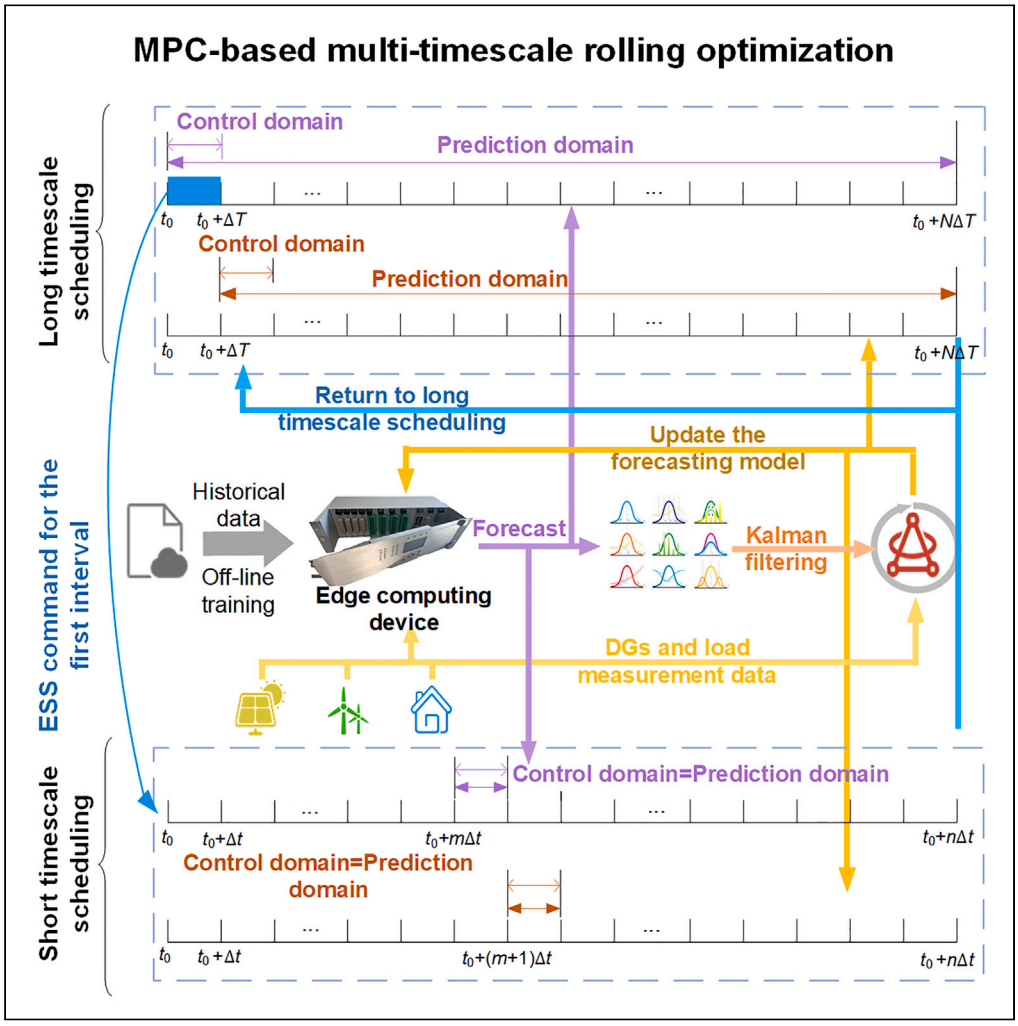
iScience

CellPress
OPEN ACCESS

Article

A lightweight method of integrated local load forecasting and control of edge computing in active distribution networks



Yubo Wang, Xingang Zhao, Kangsheng Wang, He Chen, Yang Wang, Hao Yu, Peng Li

wangkangsheng@tju.edu.cn

Highlights

The adaptive sparse integration method can reduce overall model scale

The auto-encoder is designed to reduce computational and storage burden

An adaptive correction method based on the Kalman filter is proposed

The entire forecasting and optimization process is implemented on the edge side

Wang et al., iScience 27, 110271
August 16, 2024
https://doi.org/10.1016/j.isci.2024.110271

iScience

CellPress
OPEN ACCESS

Article

A lightweight method of integrated local load forecasting and control of edge computing in active distribution networks

Yubo Wang,[1,2] Xingang Zhao,[1] Kangsheng Wang,[3,6,*] He Chen,[2] Yang Wang,[4,5] Hao Yu,[5] and Peng Li[5]

SUMMARY

The strong resource constraints of edge-computing devices and the dynamic evolution of load characteristics put forward higher requirements for forecasting methods of active distribution networks. This paper proposes a lightweight adaptive ensemble learning method for local load forecasting and predictive control of active distribution networks based on edge computing in resource constrained scenarios. First, the adaptive sparse integration method is proposed to reduce the model scale. Then, the auto-encoder is introduced to downscale the model variables to further reduce computation time and storage overhead. An adaptive correction method is proposed to maintain the adaptability. Finally, a multi-timescale predictive control method for the edge side is established, which realizes the collaboration of local load forecasting and control. All cases can be deployed on an actual edge-computing device. Compared to other benchmark methods and the existing researches, the proposed method can minimize the model complexity without reducing the forecasting accuracy.

INTRODUCTION

Motivation

The digital transformation and upgrading of power system is recognized as the potential solution for improving renewable energy penetration and reducing carbon emissions.[1] As a key platform for the integration and efficient utilization of distributed generators (DGs), active distribution network (ADN) plays an important role in the development of modern energy systems,[2] whereas many new characteristics have also emerged. The new elements such as electric vehicles and DGs increase the peak-to-valley difference of loads and make the load profile more volatile and complex.[3,4] To cope with these new characteristics, the concepts such as microgrid (MG)[5] and virtual power plant (VPP)[6] have been developed in ADNs. Novel flexible power electronics such as soft open point (SOP)[7,8] have also been applied to improve renewable energy accommodation. The coordination of these flexible resources makes the operation of active distribution networks more challenging.[9]

Rapid development of digital technologies provides the opportunity for the flexible operation of active distribution networks.[10] The digital transformation of the physical power grid improves the observability and controllability of complex distribution systems.[11] A large number of digital terminals provide data support to achieve full-dimensional state sensing and optimized control decision.[12] However, the huge amount of data generated by various terminals significantly challenges the conventional centralized operation manner, facilitating the application of new digital technologies such as edge computing.[13] In the context of ADNs, edge computing refers to computation occurring at the point where data are generated, near electrical equipment and users. It should be noted that the edge in ADNs is not confined to geographical boundaries. Various applications in ADNs that deploy and utilize computing resources in close proximity to the data source fall within the category of edge computing. Compared to conventional equipment such as relay protection devices, edge computing in ADNs maintains the advantages of rapid response while offering enhanced intelligence and flexibility.

As a new computing paradigm, edge computing aims to migrate operational applications to the edge side,[14] thus satisfying the key requirements such as fast response[15] and privacy protection,[16] and finally achieving local intelligent operation with the ability of autonomous forecasting and control.[17,18] Wang et al.[19] proposed a flexible control paradigm composed of a cloud platform and digital signal processor (DSP) chips to realize dynamic economic dispatch for a remote MG. The paradigm applied the DSP chip of the inverter in an arbitrary renewable generator for real-time computation to make decision instructions. Li et al.[20] established a unified energy management framework to realize sustainable operation of various renewable energy sources in microgrids and improve energy utilization. Munir et al.[21] formulated

[1]North China Electric Power University, Beijing 102206, China
[2]Beijing SmartChip Microelectronics Technology Company Limited, Beijing 102200, China
[3]State Grid Nantong Power Supply Company, Jiangsu 226001, China
[4]State Grid Shanghai Municipal Electric Power Company, Shanghai 200122, China
[5]Key Laboratory of Smart Grid of Ministry of Education, Tianjin University, Tianjin 300072, China
[6]Lead contact
*Correspondence: wangkangsheng@tju.edu.cn
https://doi.org/10.1016/j.isci.2024.110271

a risk-aware energy scheduling method based on multi-agent deep reinforcement learning for the microgrids to reduce the risk of energy shortfall caused by the volatilities of both energy consumption and generation. These studies demonstrated the promising and significant role of edge computing technology in the energy sector.

As a key prerequisite to support local autonomous operation at the edge side,[22] the accuracy of load forecasting is a crucial issue.[23,24] However, edge-side forecasting is quite different from conventional load forecasting of bulk power systems, which mainly focuses on the aggregated load at the system level.[25] Due to the smaller territory coverage of an edge-computing device, the variation and uncertainty of local loads are intensified, which increases the difficulty in load forecasting on the edge side.

Moreover, the edge-side load forecasting method needs to take into account both accuracy and algorithm complexity. An edge-computing device usually has limited computation and storage resources.[26] However, as a local service center, it must conduct multiple grid services such as protection, control, and monitoring simultaneously. The constraints on the resources occupied by a single service are more stringent, which puts higher requirements on the algorithm design of edge-side load forecasting.[27] In this context, the forecasting model for edge computing not only pursues high accuracy, but also must minimize computational and storage resources to be deployed on edge-computing devices.

In addition, it is challenging to maintain the adaptivity of the forecasting model in the long-term operation. The locations of the loads managed by an edge-computing device are generally close to each other, and the composition is relatively monotonous. The long-term influence factors, such as changes in seasons, climates, and energy consumption habits, could have a similar effect on the overall loads, which makes the evolution of load characteristics more significant. For example, the characteristics of the load profile usually differ during the summer, winter, and seasonal transition periods. The gradual evolution of demand restricts the continuable applicability of a single model in load forecasting. It is also difficult to train a unified model that is suitable for all scenarios because there is no guarantee that the training data over a long period share the same characteristic. An effective adaptive update mechanism must be established for the forecasting model on the edge side.

Literature survey

Ensemble learning models and strategies

Ensemble techniques method trains a series of identical or dissimilar models to learn knowledge and integrates the output of each model based on specific rules. A series of researches have revealed that ensemble forecasting can make full use of the advantages of the base predictors and lower the risk of falling into local optimality. For example, the internal corrosion rate for oil and gas pipeline was forecasted by four ensemble learning approaches in,[28] namely random forest (RF), gradient boosting regression tree (GBRT), adaptive boosting (AdaBoost), and extreme gradient boosting (XGBoost). Mohammad et al.[29] evaluated ensemble learning methods (bagging, boosting, and modified bagging) potential in forecasting microbially induced concrete corrosion in sewer systems from the data mining perspective. They both demonstrated that ensemble learning model has a higher accuracy than a single model.

The ensemble strategies of relevant studies are categorized and summarized in Table 1. As shown in Table 1, the ensemble models can be generally divided into homogeneous ensembles founded on the same base predictors and heterogeneous ensembles founded on various base predictors. Homogeneous ensembles can be further classified into parallel and sequential ensembles. In parallel ensembles, homogeneous base learners are trained independently and in parallel and then are combined by simple averaging. For example, a rolling decomposition-ensemble model based on support vector regression (SVR) was proposed for quarterly gasoline consumption forecasting in China in a study by Yu et al.[30] Typical algorithms include bagging and RF. On the other hand, sequential ensembles train homogeneous base predictors sequentially in an adaptive way and then combine them through weighted averaging. The representative algorithm is AdaBoost, where the distribution of the training set is adaptively adjusted according to the results of the previous base predictor. In heterogeneous ensembles, base predictors are different types of machine learning models. Stacking is the most widely used heterogeneous ensemble framework, predictors of various types are trained in parallel, and their outputs are then fed as inputs to a second layer. So the stacking ensemble model focuses on how to develop an efficient ensemble strategy.

The current ensemble strategies mainly include the simple addition strategy, weighted least square method, and heuristic evolutionary algorithm. Most studies[31,32,35,36] consider the base predictors to be equally important, and by simple average weighting to eliminate random errors, this often produce a strong model with less bias. However, the contribution of the base predictor cannot be measured, which limits further improvements in the accuracy of the ensemble model. To address this problem, some scholars have proposed different ensemble strategies to minimize errors. For example, an improved multi-objective particle swarm optimization (MOPSO) algorithm integrated with a dynamic heterogeneous mutation operator was designed to forecast Baltic Dry Index (BDI) in a study by Li et al.[33] A novel multi-objective Mayfly algorithm was proposed to estimate the optimal weight coefficients for integrating the forecasting values of the subseries in a study by Liu et al[34]; while Liu and Yang[37] and Duan et al.[41] adopted multi-objective wolf colony algorithm, and deep belief networks based on PSO, respectively, to determine the weights of the ensemble forecasting model.

Although there are countless ensemble algorithms developed and applied in different scenarios, most of the existing research focuses on how to improve the accuracy without considering the complexity of the algorithms and the collaboration of forecasting and control. Since homogeneous ensembles train the same type of base predictors, they tend to have smaller computational overhead and are more suitable for edge computing scenarios. In addition, when the data distribution of the loads changes slowly, most of the studies obtain new forecasting models by retraining. However, frequent retraining requires a large computational overhead, and there is a lack of scientific judgment on how to select the new training set. Therefore, there is an urgent need for a low-complexity adaptive updating algorithm to achieve long-term effective forecasting of edge-side loads.

Table 1. Ensemble strategies review

Literatures	Framework	Objects	Base predictors	Ensemble strategies
Liborio et al.[31]	Stacking	Bond strength of the corroded reinforced concrete	Convolution neural networks (CNNs), SVR, gradient boosting (GB), artificial neural networks (ANNs)	Stacking ensemble
Liu et al.[32]	Stacking	Fuel properties	CNNs, SVR, RF, extremely randomized trees, light gradient boosting	Stacking ensemble
Li et al.[33]	Weighting ensemble	Baltic Dry Index (BDI)	Autoregressive integrated moving average (ARIMA), extreme learning machine (ELM), exponential smoothing, SVR, ANNs, RF	Improved MOPSO algorithm
Liu et al.[34]	Weighting ensemble	Wind energy	ARIMA, ANNs, ELM et al.	Multi-objective mayfly algorithm
Ramon et al.[35]	Stacking	Wind energy	K-nearest neighbor (KNN), partial least-squares regression (PLS), ridge regression (RIDGE), SVR	Stacking ensemble
Yu et al.[30]	Addition ensemble	Gasoline consumption	SVR	Simple addition strategy
Yu et al.[36]	Addition ensemble	Monthly biofuel production	Long short-term memory (LSTM), ELM	Simple addition strategy
Liu et al.[37]	Weighting ensemble	Air quality index	Outlier robust extreme learning machine	Multi-objective wolf colony algorithm
Zhu et al.[38]	Boosting	Credit risk	Decision tree (DT), Random subspace	Multi-boosting approach
Liu et al.[39]	Addition ensemble	Wind speed	Complementary ensemble empirical mode decomposition, LSTM	Simple addition strategy
Sinvaldo et al.[40]	Addition ensemble	Wind speed	Variational mode decomposition (VMD), ARIMA, singular spectral analysis (SSA)	Simple addition strategy
Duan et al.[41]	Weighting ensemble	Wind power	VMD, LSTM	Deep belief networks based on PSO

Load forecasting for edge computing

The literatures on load forecasting for edge computing can be mainly divided into two categories. The first category uses cloud-edge collaboration as the overall framework, with models trained in the cloud and then distributed down to the edge side for forecasting. For example, an ensemble learning method based on least squares support vector machines (LSSVMs) was proposed in a study by Deng et al.[42] for forecasting the cooling load of buildings under different operating conditions, providing a decision basis for the optimal control of chilled water systems. Zhou et al.[43] employed extreme learning machine (ELM) and feature selection to build an energy consumption forecasting model for edge servers and validates it in terms of accuracy and training time under three different operating conditions. In a study by Hou et al.,[44] an anomaly-aware cloud-side collaborative forecasting approach was proposed considering anomalous data, in which the offline model was trained on the cloud and the online model was run in real-time on the edge side. Zhong et al.[45] also adopted a similar cloud-edge collaborative

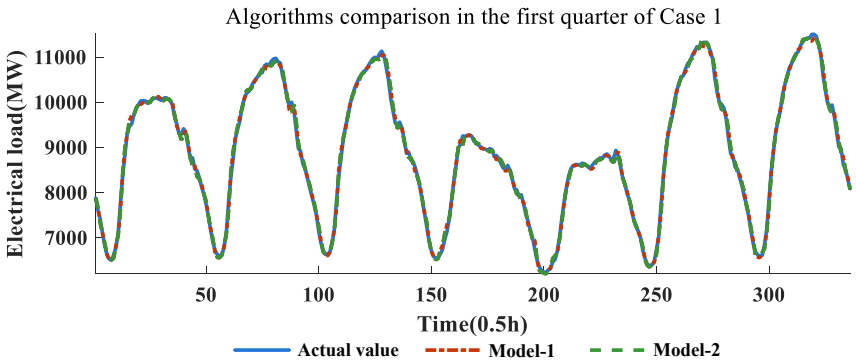


Figure 1. Comparison of the forecasting results in case 1

Table 2. Comparison of forecasting results in case 1

	Indicators	Bi-LSTM-Attention	Model-1	Model-2
First quarter	MAPE	1.17	0.64	**0.64**
	RMSE	138.10	75.24	**75.22**
Second quarter	MAPE	1.61	0.81	**0.81**
	RMSE	184.9	99.54	99.56
Third quarter	MAPE	1.54	0.73	**0.73**
	RMSE	182.50	88.91	88.94
Fourth quarter	MAPE	1.22	0.76	**0.76**
	RMSE	126.60	81.67	**81.67**

framework for user-level load forecasting, and the training of the sequence to sequence-LSTM was also conducted on the cloud. Li et al.[46] proposed a federal learning-based ultra-short-term load forecasting method with model parameters generated collaboratively by each edge node to improve accuracy. In a study by Luo et al.,[47] a stacked model based on AE was built on the edge computing platform to forecasting electric vehicles load.

However, the implementation environment constraint and algorithm complexity are not fully considered in existing studies about cloud-edge collaborative forecasting approaches. Most researches utilized virtual edge-computing environment simulated on computers, and may have difficulty to be implemented on real edge-computing devices that lacks supporting environment for complex deep learning algorithms. Meanwhile, the cloud-trained offline model needs to be retrained frequently to adapt to the dynamic changes in load characteristics, which limits the long-term adaptivity of cloud-edge approaches, and increases the computational and communicational burden.

The second category carries out training and forecasting all at the edge side, which fits better with the practical needs of edge computing, but requires a balance between accuracy and complexity. Han et al.[48] compared the complexity and accuracy of several lightweight neural networks on a Raspberry Pi device. Deb et al.[49] also proposed an LSTM-based forecasting model and tested it on a Raspberry Pi. However, their results showed that the CPU and RAM occupancy of Raspberry Pi reached 30.24% and 34.61% when training, and 26.03% and 40.36% when forecasting. Moreover, the complexity of single-step forecasting method may increase exponentially in day-ahead forecasting with sequential time slots. In this case, the resources of dedicated edge-computing devices in distribution networks may be insufficient. In this context, feedforward neural networks that do not require weight iterations are more suitable as the base algorithm. For example, an ELM method was utilized for hourly forecasting of photovoltaic-assisted charging stations in a study by Shi et al.[50] An online learning scheme based on echo state network (ESN) was designed for minute-level forecasting of residential load in a study by Fujimoto et al.[51] However, these studies mainly focus on single-step forecasting and are not suitable to support the local predictive operation of distribution networks.

In summary, the majority of existing research relies on cloud platforms with ample computing and storage resources. However, there are relatively few studies that implement both training and inference of forecasting on edge devices. Research conducted on edge devices typically exhausts all computing resources for forecasting and does not support multi-step and multi-granularity forecasting. This limitation hinders further predictive operations at the local edge level of ADNs.

Contributions

In this study, a lightweight adaptive ensemble learning method for edge computing is proposed to address the challenges in the edge-side load forecasting and predictive operation, which minimizes the model complexity without reducing accuracy. The main contributions are summarized as follows.

(1) A novel ensemble learning forecasting method for edge computing of ADNs is proposed. The ESN is utilized as the lightweight algorithm. The TrAdaBoost framework is adopted to handle diversity in load characteristics, in which an adaptive sparse integration method is established to reduce the overall model scale. The searching process of key base predictors under the new framework is modified to support multi-step and multi-granularity forecasting. Additionally, an AE is designed to downscale the model variables further, reducing computational and storage burden.

Table 3. Comparison of the different models in case 2 (MAPE, %)

Region Number	GBDT	AdaBoost.R2	STA-TAB.R2	Model-1	Model-2	Model-4
1	25.09	24.86	18.01	13.31	13.30	**12.85**
11	25.15	23.54	17.53	10.38	10.38	**10.31**
12	25.09	24.34	19.80	8.04	8.05	9.32
15	22.12	22.02	18.96	14.88	14.88	**14.76**
20	19.43	20.08	16.52	11.02	11.02	12.05

Table 4. Comparison of calculation time of different models in case 2

	GBDT	AdaBoost.R2	STA-TAB.R2	Model-1	Model-2	Model-4
Training time/s	0.20	17.03	1.21	17.76	18.92	**18.92**
Forecasting time/s	0.03	0.14	0.07	0.14	0.02	**0.47**

(2) An adaptive correction method based on the Kalman filter with modified covariance is proposed to adjust the model with short-term measured data, significantly improving the speed and accuracy of adaptive forecasting. The output matrix is decomposed into multiple independent state vectors and corrected separately, allowing simultaneous updates for multi-step forecasting. Additionally, a model weight reallocation mechanism is developed to enhance the adaptivity and long-term performance of the integrated forecasting model.

(3) A multi-timescale predictive control method for the edge side is established based on local forecasting capability, enabling collaboration between local load forecasting and control within the model predictive control (MPC) framework. The entire training, reasoning, and optimization process is implemented on the edge side. The proposed method offers a feasible solution for autonomous and intelligent operation of a local area within active distribution networks using edge-computing devices.

RESULTS

To demonstrate the effectiveness of the proposed model, three benchmark models were compared as follows:

Model-1 (conventional integrated model): ESN is used as the base predictor and TrAdaBoost is used as the ensemble learning framework, with no subsequent model updates.

Model-2 (sparse integration model): compared to Model-1, the sparse integrated forecasting model is used for sparse, and the model is not updated subsequently.

Model-3 (sparse adaptive model): the initial model is the same as Model-2, and Model-3 is updated daily using the adaptive correction algorithm.

Model-4 (the proposed model): compared to Model-3, the model weights are reallocated during long-term forecasting.

In addition, the proposed model in this paper is compared with the traditional machine learning model, which uses the gradient boosting decision tree (GBDT), and the traditional ensemble learning model, which uses the AdaBoost algorithm (AdaBoost.R2).[52] The evaluation criteria used in this study are mean absolute percentage deviation (MAPE) and root-mean-square error (RMSE), which are calculated as follows:

$$MAPE = \sum_{i=1}^{N_T} \frac{1}{N} \left| \frac{x_i - \hat{x}_i}{x_i} \right| \times 100\% \quad \text{(Equation 1)}$$

$$RMSE = \sqrt{\sum_{i=1}^{N_T} \frac{1}{N} (x_i - \hat{x}_i)^2} \quad \text{(Equation 2)}$$

where $\hat{x}_i$ is the forecasting value for the i th hour and x_i is the actual value for the i th hour, N_T is the number of time intervals.

The proposed forecasting algorithm was tested on a real edge-computing device named WFDT-7000, equipped with a FUXI-H2 32-bit 4-core CPU running at 800 MHz and 1 GB of RAM.[53] To enable a quick comparison of accuracy and computation time for each benchmark model, case 1 and case 2 were executed on MATLAB 2018, utilizing an Intel Core i7 64-bit 8-core CPU operating at 2.9 GHz, 16 GB of RAM, and 500 GB of data storage. Case 3 was deployed on a Raspberry Pi-based edge computing platform, featuring a Cortex-A72 64-bit 4-core CPU at 1.5 GHz, 4 GB of RAM, and 32 GB of data storage. Additionally, the Raspberry Pi includes two USB 2.0 ports and two USB 3.0 ports, supporting various network interfaces such as Gigabit Ethernet, 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, and Bluetooth 5.0 BLE. It is evident that, in terms of computational performance, PCs significantly outperform Raspberry Pi and the edge-computing device.

Table 5. Comparison of the different models after expanding the training set (MAPE, %)

Region Number	GBDT	AdaBoost.R2	STA-TAB.R2	Model-1	Model-2	Model-4
4	20.33	19.85	17.80	5.03	5.03	6.02
7	22.01	22.02	18.96	7.42	7.42	**6.43**
8	22.46	21.89	18.92	6.92	6.92	**6.12**
20	22.01	21.57	20.11	6.54	6.53	6.94

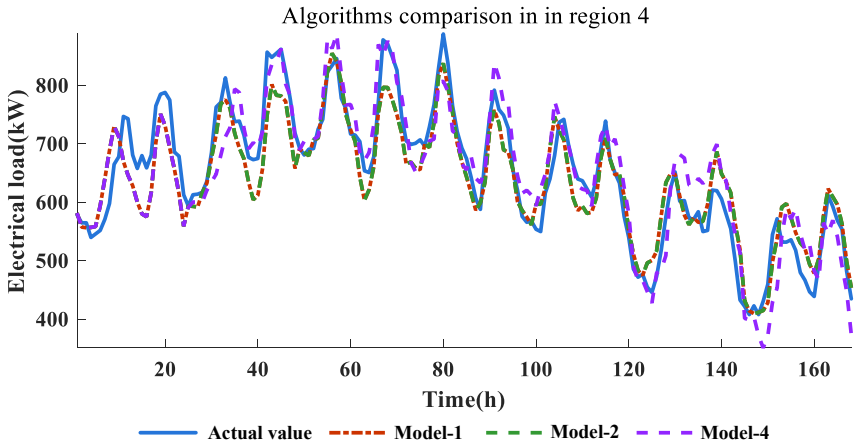


Figure 2. Comparison of the different models in region 4

Case 1: NSW load dataset (hourly single-step forecasting)

The model proposed in this paper focuses on multi-step forecasting, but it is also applicable for single-step forecasting. Case 1 uses the New South Wales (NSW) load dataset as the experimental subject and compares the forecasting results with a study by Wang et al.[54]

All conditions were set to be consistent with 54 to ensure fairness. The single-step forecasting data are updated quickly and can correct the curve trend, so case 1 is not forecasted online. Input related parameters include: $n_0 = 30$, leaking parameter candidate set $\{0.92, 0.94, 0.96, 0.98\}$, reservoir pool size candidate set $\{700, 800, 900, 1000, 1100, 1200\}$, regularization parameter candidate set $\{10, 10^2, 10^3, 10^4\}$, $N_0 = 80$, $D_L = 100$. The remaining training samples are divided into source and target domains by 9:1.

A comparison of the forecasting results for the first quarter is shown in Figure 1. It can be seen that the forecasting accuracies of both Model-1 and Model-2 are high.

The specific statistics of the forecasting results are listed in Table 2. It can be seen that the accuracies of Model-1 and Model-2 are comparable and both are higher than the deep learning algorithm proposed in 54 (abbreviated as Bi-LSTM-Attention). However, the number of base predictors of Model-2 is 12, while Model-1 is 80. The integration scale of Model-2 is substantially reduced, which is more relevant for edge computing.

Case 2: Global Energy Forecasting Competition 2012 (day-ahead forecasting)

Training sets with small data samples

Case 2 uses the Global Energy Forecasting Competition 2012 load dataset as the experimental subject and compares the forecasting results with a study by Xu et al.[55] (abbreviated as STA-TAB.R2). The dataset consists of load data from 20 regions in the United States, including different service attributes such as residential areas and commercial buildings. All conditions were set to be consistent with 55 to ensure fairness. Input related parameters include: $n_0 = 20$, $\Gamma = 70$, leaking parameter candidate set $\{0.92, 0.94, 0.96, 0.98\}$, reservoir

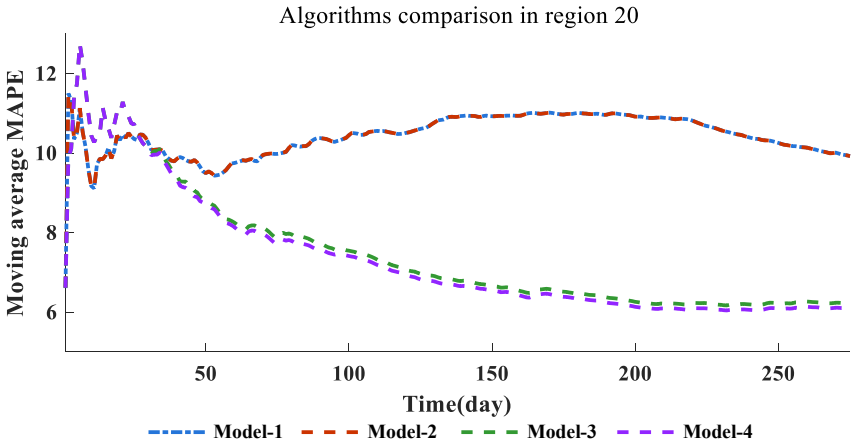


Figure 3. Comparison of the different models for region 20

Table 6. Comparison of different models for long-term forecasting (MAPE, %)

Region Number	Model-1	Model-2	Model-3	Model-4
12	8.42	8.42	6.51	**6.43**
20	9.54	9.56	6.25	**6.19**

pool size candidate set {700,800,900,1000,1100,1200}, regularization parameter candidate set $\{10,10^2,10^3,10^4\}$, $N_0 = 80$, $D_L = 100$, $M_d = 50$, $N_d = 20$, $k = 30$, $T_L = 30$. The initial value of $\{\boldsymbol{P}\}$ and $\{\boldsymbol{Q}\}$ take 10^{-2} and 10^{-4} for the main diagonal element and 0 for the rest; the initial value of each element of $\boldsymbol{R}_M$ takes 10^{-4}.

All models were subjected to 10 independent experiments, and the average value was taken as the final result. A comparison of different models for each region is shown in Table 3.

Since the test set is only 7 days long and the model weights are not adjusted, Model-4 has the same results as Model-3. As can be seen from Table 3, GBDT has the worst forecasting results, followed by AdaBoost.R2. Model-4 has the best forecasting performance on regions 1, 11, and 15, but does not achieve the best performance on regions 12 and 20, due to the short test set time and non-ideal values of system noises and measurement noises.

A comparison of the computation times of different models is shown in Table 4, where GBDT has the shortest training time and forecasting time due to the fact that all other models use ensemble learning, but accordingly, GBDT has the worst forecasting performance.

Since Model-1 uses TrAdaBoost, the training time is slightly longer compared to AdaBoost. Model-2 and Model-4 additionally consider a sparse integration mechanism and have the longest training time, but the increased computational overhead is extremely low. In terms of forecasting time, Model-4 corrects the model daily, so it has the longest forecasting time cost, but Model-4 does not need to be retrained for subsequent forecasting. Since the forecasting time is much less than the training time, Model-4 has a significant advantage in terms of computational overhead.

Training set with larger data samples

In this section, the number of training samples in the training set is expanded from 90 to 365, and the regions are replaced with 4, 7, 8, and 20. The comparison of different models in each region is shown in Table 5, and the comparison of different models in region 4 is shown in Figure 2. From Table 5, it can be seen that GBDT still has the worst forecasting results, followed by AdaBoost.R2, which is consistent with the results of the previous analysis. Model-4 has the best performance on regions 7 and 8, but does not achieve the best performance on regions 4 and 20. From the results of region 20, it is clear that expanding the number of samples in the dataset can significantly improve the accuracy of ensemble learning model.

k-step ahead forecasting: based on the aforementioned training and testing sets, the forecasting errors for 24-step, 12-step, 6-step, and single step forecasting of region 20 are 6.94, 5.32, 3.27, and 0.64, respectively. Due to the latest measurement data being able to calibrate the model faster, smaller forecasting steps result in smaller forecasting errors.

Long-term forecasting analysis

To verify the adaptability of Model-3 and Model-4, additional experiments were conducted for regions 12 and 20 where the best prediction results were not achieved in the short term. The other conditions were the same as before, only the test set time was extended.

The advantages of the ensemble learning model are obvious; the analysis focuses on Model-1 to Model-4 for a clear comparison. The daily moving average MAPE curves of region 20 from 2007.4.1 to 2007.12.31 are shown in Figure 3. The specific statistical errors are shown in Table 6.

As can be seen from Figure 3 and Table 6, Model-3 and Model-4 can guarantee the accuracy of forecasting results without retraining in long-term forecasting, because the models are updated every day. After one month, the performance of Model-3 and Model-4 then steadily outperforms that of Model-1 and Model-2. Model-4 still has a small improvement compared to Model-3 because the model weight of base predictor is also reassigned every month.

Case 3: DKASC photovoltaic dataset

The model proposed in this paper focuses on load forecasting, but it is also applicable for renewable generation forecasting. Case 3 uses the Australian DKASC PV dataset as the experimental subject. Considering the characteristics of PV power generation, the daily time period of

Table 7. PV power forecasting results for different granularity (MAPE, %)

Forecasting granularity	Model-1	Model-2	Model-4
1h-level	3.80	3.80	3.38
15min-level	10.11	10.15	9.97
5min-level	14.84	14.97	14.75

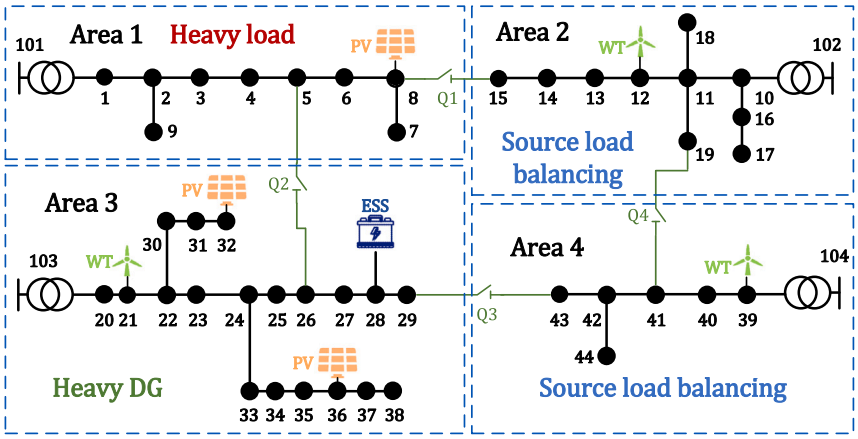


Figure 4. Topology of a practical distribution network in Tianjin
See also Tables S1 and S2.

7:00 to 19:00 was selected for forecasting in 2019. Historical data from January 1 to November 31 (excluding days with missing PV output data) in Site 33 was used as the training set, and historical data from December 1 to December 7 was used as the test set.

Renewable generation exhibits higher uncertainty and lower timing compared to load. Environmental variables such as solar irradiance and temperature can have an impact on photovoltaic power generation, but different environmental variables have different contributions and impacts. Therefore, ELM, which is similar to the structure of ESN and has stronger fitting effect, is selected as the base predictor. Before inputting environmental variables into the proposed method, correlation analysis is conducted to screen out the main factors. Then principal component analysis (PCA) is employed to further extract meteorological factors highly correlated with photovoltaic power. The specific steps are as follows:

Step 1: correlation analysis of the influencing factors of PV power generation is carried out, and the Pearson correlation coefficient is used to screen out the main factors. Then, the meteorological factor matrix is obtained, whose dimensions are the number of samples and the number of main factors, respectively.

Step 2: PCA was used to further extract the meteorological factors associated with high PV output. The principal components with a contribution rate greater than 90% were selected as the integrated meteorological factors and used as model inputs.

The day-ahead PV power forecasting results for different granularity are shown in Table 7. At all levels of granularity, the forecasting performance of Model-4 is higher than that of Model-1 and Model-2. Meanwhile, the finer granularity leads to a lower accuracy due to the fact that the DG fluctuates more drastically in the short term. Combining case 1 and case 2 shows that the accuracy decreases as the dimensionality of the output of the day-ahead forecasting increases, for both load and DG.

Case 4: Multi-timescale local predictive control

Since the proposed lightweight forecasting algorithm can be deployed on the edge-computing device and supports multi-timescale forecasting, this enables the edge-computing device to realize local predictive control. Case 3 is analyzed to demonstrate the value of the application in the autonomous and intelligent operation of a local area using edge-computing devices.

In this section, a demonstration area of distribution network in Tianjin, China, is selected for analysis. The topology is shown in Figure 4, and the detailed parameters are shown in Tables S1 and S2. The voltage levels in the demonstration area are all 10.5 kV, and the active power and

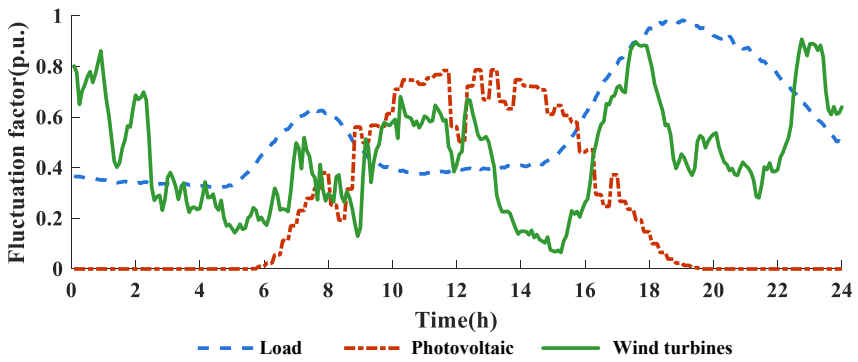


Figure 5. Daily output curves of DGs

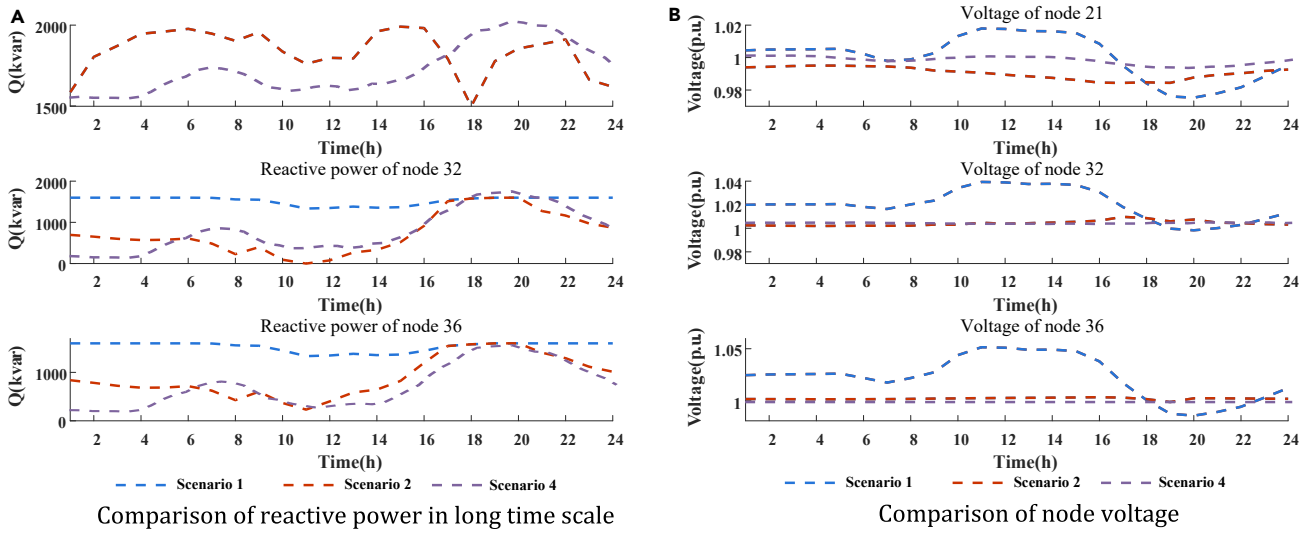


Figure 6. Scheduling results on long time scales

(A) Comparison of reactive power in long time scale.

(B) Comparison of node voltage.

reactive power loads are 9.99 MW and 7.34 Mvar, respectively. In order to fully consider the impact of high penetration DG integration on the distribution network, three groups of photovoltaic and three wind turbines are connected respectively, and the penetration rate of DGs reaches 90.11%. No reduction of power output from distributed power sources is allowed. The rated capacity of energy storage system (ESS) is 2

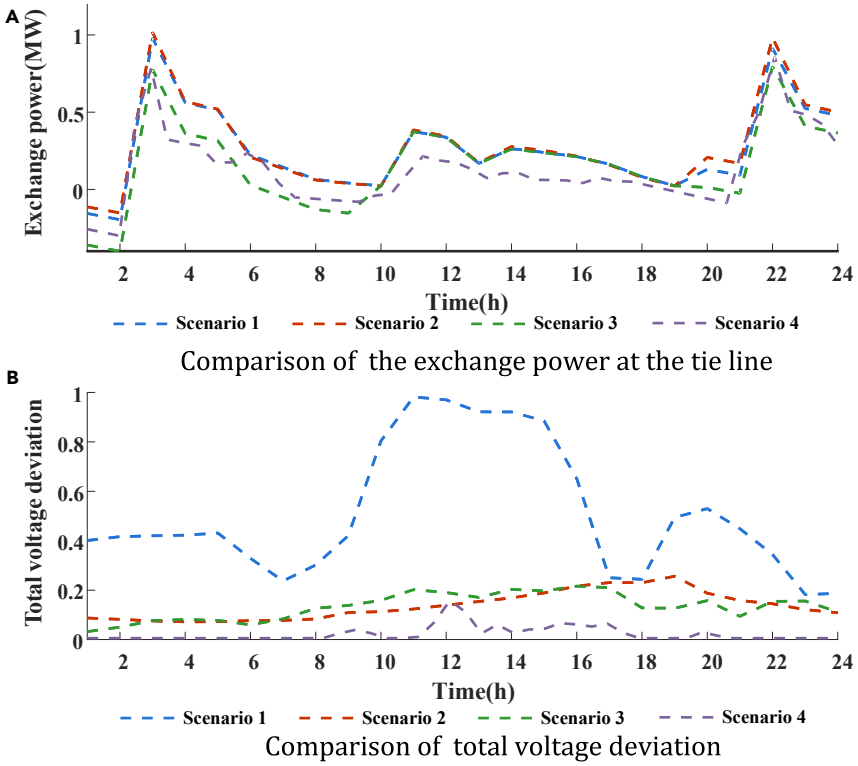


Figure 7. Long timescale optimization results in different scenarios

(A) Comparison of the exchange power at the tie line.

(B) Comparison of total voltage deviation.

Table 8. Comparison of optimization results in different scenarios

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Total voltage deviation	12.20	3.28	3.00	2.45
Total network loss/MWh	1.96	1.08	1.04	1.00

MW; the charging and discharging power is limited to 0.5 MW/h. Input related parameters include: $SOC^{ESS,min} = 10\%$, $SOC^{ESS,max} = 90\%$, $\Delta T = 1h$, $\Delta t = 5$ min.

The following three scenarios are selected to analysis and verify the effectiveness of the proposed local control strategy.

Scenario 1: no control of DG and ESS is not considered.

Scenario 2: optimizing only the reactive power of DG without considering ESS.

Scenario 3: optimizing both the reactive power of DG and ESS.

Scenario 4: close all isolation switches and perform centralized optimization scheduling.

To visually compare the effectiveness of the proposed method, the DG rich area with source node 103 is selected for detailed analysis. The daily active output curve of DGs is shown in Figure 5.

The scheduling results for scenario 1, scenario 2, and scenario 4 on long time scales are shown in Figure 6. It can be seen that voltage fluctuations at each node are smaller when DGs are involved in voltage control. In scenario 1, frequent voltage crossing limits occur at node 36 for some periods because DGs are not involved in voltage control.

Figure 7 gives the optimization results for different scenarios with long time scales. Figure 7A shows the contact line exchange power between the area under edge-computing device and superior grid, and Figure 7B shows the total system voltage deviation for the area under edge-computing device.

Table 8 lists the results of total system voltage deviation and network loss for different scenarios in the typical day. It can be seen that the introduction of ESS can significantly reduce network loss and decrease voltage deviation without considering the dispatch cost of ESS.

The optimization results of scenarios 1, 3, and 4 for short time scales are given in Figure 8. For fairness, scenario 1 also considers the ESS, and the charging and discharging strategy is the same as that of scenario 3. From Figure 8, it can be seen that if the DGs are not controlled, the system network loss is large and frequent voltage crossing limits will occur. Multi-timescale local control can have a significant loss reduction effect. The integration of the multi-timescale predictive control method on edge-computing devices can significantly improve the economy and safety of edge-side operation.

In addition, we can also know that centralized optimization has achieved the best optimization results. However, centralized approaches may be infeasible in practice due to communication limitations and the availability of data resources. In particular, in distribution networks, DGs are typically widespread and of small scale, making monitoring and control from a central station challenging. As shown in Figure 8, the proposed method can obtain results that are close to those of centralized methods. This serves to demonstrate the advantage of the proposed method in achieving quasi-optimal control performance without the need for centralized communication, global information, and large-scale optimization computation.

DISCUSSION

This paper proposes a lightweight adaptive ensemble learning method for the load forecasting on edge-computing devices, which is then coordinatively employed in the multi-time scale predictive control to support the local intelligent operation based on edge computing in active distribution networks. The following conclusions can be drawn from the study.

First, to address the unstable performance of a single ESN, an adaptive sparse integration method is proposed to reduce the integration scale by removing non-critical base predictors. Then, AE is introduced to decrease the variables dimensionality, further reducing the

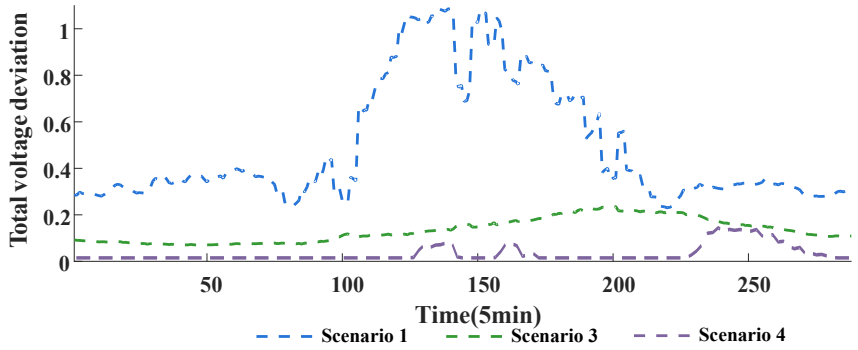


Figure 8. Voltage deviations of scenarios 1 and 3 for short time scales

computation and storage resource overhead during online updating. Simulation experiments on two practical datasets show that compared with the existing study, the hourly and day-ahead forecasting accuracy (measured by MAPE) increased by at least 37.70% and 22.15%, respectively.

Second, to address the adaptation of the proposed model, the Kalman filter with corrected covariance is employed to daily correct the model parameters. The proposed online forecasting method can minimize the model complexity without reducing the forecasting accuracy. In particular, AE can significantly reduce the computational and storage overhead. The storage overhead is only 1% of the model without AE. The results show that the performance of the proposed method is better than that of the three benchmark models in long-term forecasting.

Finally, a multi-time scale dynamic optimal scheduling method based on MPC is proposed to realize long timescale economic operation and short timescale safe operation. The results show that the proposed method can reduce network loss by 46.93% and significantly improve the voltage profile compared to the no-control strategy. In terms of hardware platform, both training and forecasting of the programming can be implemented on a real edge-computing device. The local control program incorporating the forecasting information can be implemented on the Raspberry Pi-based edge platform, which demonstrates the feasibility of the algorithm on edge-computing devices.

Limitations of the study

The proposed lightweight adaptive ensemble learning method allows accurate source and load forecasting without degrading the accuracy. Nevertheless, there are several limitations and future work can augment our analysis from the following aspects.

In the future, the important meteorological data can be further focused by introducing an attention mechanism to avoid redundancy and strong correlation of input variables. Meanwhile, the actual measurement data are often missing and anomalous, so a data cleaning module can be further added to the program. In addition, cone transformation of multi-timescale optimization models can be considered to achieve faster solving and better fit the practical needs of edge computing.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Sparse integrated forecasting model under TrAdaBoost framework
  - Architecture of the echo state network
  - Integration process of TrAdaBoost
  - Process of adaptive sparse integration
  - Adaptive correction of the forecasting model
  - Flow chart of the proposed method
- MULTI-TIMESCALE LOCAL PREDICTIVE CONTROL FOR EDGE COMPUTING
  - Multi-timescale model predictive control architecture
  - A typical edge-computing application scenario
  - MPC-based multi-timescale rolling optimization
  - Flow chart of multi-time scale rolling optimization
- ADDITIONAL RESOURCES
  - Nomenclature

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.isci.2024.110271.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China (No.U22B20114) and the Scientific Research Programs for High Level Talents of Beijing Smart-chip Microelectronics Technology Co., Ltd.

AUTHOR CONTRIBUTIONS

Conceptualization, Y.W., X.Z., K.W., H.Y., and P.L.; methodology, Y.W., X.Z., K.W., and H.C.; investigation, Y.W., X.Z., K.W., H.C., and Y.W.; writing – original draft, Y.W., X.Z., K.W., and H.Y.; writing – review & editing, Y.W., X.Z., K.W., and H.Y.; funding acquisition, Y.W. and X.Z.; supervision, H.C., Y.W., and P.L.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: November 9, 2023
Revised: April 5, 2024
Accepted: June 12, 2024
Published: June 14, 2024

REFERENCES

1. Byungkwon, P., Dong, J., Liu, B., and Kuruganti, T. (2023). Decarbonizing the grid: Utilizing demand-side flexibility for carbon emission reduction through locational marginal emissions in distribution networks. Appl. Energy *330*, 120303. https://doi.org/10.1016/j.apenergy.2022.120303.
2. Sadeghian, O., Nazari-heris, M., Abapour, M., Taheri, S.S., and Zare, K. (2019). Improving reliability of distribution networks using plug-in electric vehicles and demand response. J. Mod. Power Syst. Clean Energy *7*, 1189–1199. https://doi.org/10.1007/s40565-019-0523-8.
3. Jian, J., Zhao, J., Ji, H., Bai, L., Xu, J., Li, P., Wu, J., and Wang, C. (2024). Supply restoration of data centers in flexible distribution networks with spatial-temporal regulation. IEEE Trans. Smart Grid *15*, 340–354. https://doi.org/10.1109/TSG.2023.3286844.
4. Badal, F.R., Das, P., Sarker, S.K., and Das, S.K. (2019). A survey on control issues in renewable energy integration and microgrid. Prot. Control Mod. Power Syst. *4*, 8. https://doi.org/10.1186/s41601-019-0122-8.
5. Nasser, N., and Fazeli, M. (2021). Buffered-microgrid structure for future power networks: a seamless microgrid control. IEEE Trans. Smart Grid *12*, 131–140. https://doi.org/10.1109/TSG.2020.3015573.
6. Yang, X., and Zhang, Y. (2021). A comprehensive review on electric vehicles integrated in virtual power plants. Sustain. Energy Technol. Assessments *48*, 101678. https://doi.org/10.1016/j.seta.2021.101678.
7. Zhang, S., Fang, Y., Zhang, H., Cheng, H., and Wang, X. (2022). Maximum hosting capacity of photovoltaic generation in sop-based power distribution network integrated with electric vehicles. IEEE Trans. Industr. Inform. *18*, 8213–8224. https://doi.org/10.1109/TII.2022.3140870.
8. Chen, B., Fei, W., Tian, C., and Yuan, J. (2018). Research on an improved hybrid unified power flow controller. IEEE Trans. Ind. Appl. *54*, 5649–5660. https://doi.org/10.1109/TIA.2018.2848654.
9. Azuatalam, D., CChapman, A., and Verbič, G. (2021). Probabilistic assessment of impact of flexible loads under network tariffs in low-voltage distribution networks. J. Mod. Power Syst. Clean Energy *9*, 951–962.
10. Zhou, M., Yan, J., and Feng, D. (2019). Digital twin framework and its application to power grid online analysis. CSEE J. Power Energy Syst. *5*, 391–398. https://doi.org/10.17775/CSEEJPES.2018.01460.
11. Zhao, J., Zhang, Z., Yu, H., Ji, H., Li, P., Xi, W., Yan, J., and Wang, C. (2023). Cloud-Edge Collaboration-Based Local Voltage Control for DGs With Privacy Preservation. IEEE Trans. Industr. Inform. *19*, 98–108. https://doi.org/10.1109/TII.2022.3172901.
12. Pan, J., and McElhannon, J. (2018). Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J. *5*, 439–449. https://doi.org/10.1109/JIOT.2017.2767608.
13. Feng, C., Wang, Y., Chen, Q., Ding, Y., Strbac, G., and Kang, C. (2021). Smart grid encounters edge computing: opportunities and applications. Adv. Appl. Energy *1*, 100006. https://doi.org/10.1016/j.adapen.2020.100006.
14. Tian, Z., Shi, W., Wang, Y., Zhu, C., Du, X., Su, S., Sun, Y., and Guizani, N. (2019). Real-time lateral movement detection based on evidence reasoning network for edge computing environment. IEEE Trans. Industr. Inform. *15*, 4285–4294.
15. Li, X., He, J., Vijayakumar, P., Zhang, X., and Chang, V. (2022). A verifiable privacy-preserving machine learning prediction scheme for edge-enhanced HCPSs. IEEE Trans. Industr. Inform. *18*, 5494–5503.
16. Corcoran, P., and Datta, S.K. (2016). Mobile-edge computing and the internet of things for consumers: extending cloud computing and services to the edge of the network. IEEE Consum. Electron. Mag. *5*, 73–74.
17. Guo, H., and Liu, J. (2018). Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks. IEEE Trans. Veh. Technol. *67*, 4514–4526.
18. Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: vision and challenges. IEEE Internet Things J. *3*, 637–646.
19. Wang, S., Wang, X., and Wu, W. (2020). Cloud computing and local chip-based dynamic economic dispatch for microgrids. IEEE Trans. Smart Grid *11*, 3774–3784.
20. Li, W., Yang, T., Delicato, F.C., Pires, P.F., Tari, Z., Khan, S.U., and Zomaya, A.Y. (2018). On enabling sustainable edge computing with renewable energy resources. IEEE Commun. Mag. *56*, 94–101.
21. Munir, M.S., Abedin, S.F., Tran, N.H., Han, Z., Huh, E.N., and Hong, C.S. (2021). Risk-aware energy scheduling for edge computing with microgrid: a multi-agent deep reinforcement learning approach. IEEE Trans. Netw. Serv. Manage. *18*, 3476–3497.
22. Liu, X., Xiao, Z., Zhu, R., Wang, J., Liu, L., and Ma, M. (2020). Edge sensing data-imaging conversion scheme of load forecasting in smart grid. Sustain. Cities Soc. *62*, 102363. https://doi.org/10.1016/j.scs.2020.102363.
23. Fu, J., Núñez, A., and Schutter, B.D. (2021). A short-term preventive maintenance scheduling method for distribution networks with distributed generators and batteries. IEEE Trans. Power Syst. *36*, 2516–2531.
24. Kuster, C., Rezgui, Y., and Mourshed, M. (2017). Electrical load forecasting models: a critical systematic review. Sustain. Cities Soc. *35*, 257–270.
25. Chen, Y., and Zhang, D. (2021). Theory-guided deep-learning for electrical load forecasting (TgDLF) via ensemble long short-term memory. Adv. Appl. Energy *1*, 100004. https://doi.org/10.1016/j.adapen.2020.100004.
26. Ju, J., Wang, Q., Ni, M., Hu, Y., and Li, X. (2022). An online dispatch approach for distributed integrated multi-energy system considering non-ideal communication conditions. IET Energy Syst. Integrat. *4*, 488–504.
27. Liu, Y., Yang, C., Jiang, L., Xie, S., and Zhang, Y. (2019). Intelligent edge computing for IoT-based energy management in smart cities. IEEE Network *33*, 111–117. https://doi.org/10.1109/MNET.2019.1800254.
28. Mohamed, E., Daniel, H., and Mikhail, Z. (2022). Prediction of the internal corrosion rate for oil and gas pipeline: Implementation of ensemble learning techniques. J. Nat. Gas Sci. Eng. *99*, 104425.
29. Mohammad, Z., Dietmar, S., Matthias, B., and Reinhard, H. (2020). Ensemble data mining modeling in corrosion of concrete sewer: A comparative study of network-based (MLPNN & RBFNN) and tree-based (RF, CHAID, & CART) models. Adv. Eng. Inf. *43*, 101030.
30. Yu, L., Ma, Y., and Ma, M. (2021). An effective rolling decomposition-ensemble model for gasoline consumption forecasting. Energy *222*, 119869.
31. Liborio, C., Mohammad, S., Constantinos, C., Danial, J., Dmitrii, V., and Panagiotis, G. (2022). Convolution-based ensemble learning algorithms to estimate the bond strength of the corroded reinforced concrete. Construct. Build. Mater. *359*, 129504.
32. Liu, R., Liu, Y., Duan, J., Hou, F., Wang, L., Zhang, X., and Li, G. (2022). Ensemble learning directed classification and regression of hydrocarbon fuels. Fuel *324*, 124520.
33. Li, J., Hao, J., Feng, Q., Sun, X., and Liu, M. (2021). Optimal selection of heterogeneous ensemble strategies of time series forecasting with multi-objective programming. Expert Syst. Appl. *166*, 114091.
34. Liu, Z., Jiang, P., Wang, J., and Zhang, L. (2021). Ensemble forecasting system for short-term wind speed forecasting based on optimal sub-model selection and multi-objective version of mayfly optimization algorithm. Expert Syst. Appl. *177*, 114974.
35. Ramon, G., Matheus, H., Sinvaldo, R., Viviana, C., and Leandro, D. (2021). A novel decomposition-ensemble learning framework for multi-step ahead wind energy forecasting. Energy *216*, 119174.
36. Yu, L., Liang, S., Chen, R., and Lai, K.K. (2022). Predicting monthly biofuel production using a hybrid ensemble forecasting methodology. Int. J. Forecast. *38*, 3–20.

37. Liu, H., and Yang, R. (2021). A spatial multi-resolution multi-objective data-driven ensemble model for multi-step air quality index forecasting based on real-time decomposition. Comput. Ind. *125*, 103387.
38. Zhu, Y., Zhou, L., Xie, C., Wang, G.J., and Nguyen, T.V. (2019). Forecasting SMEs' credit risk in supply chain finance with an enhanced hybrid ensemble machine learning approach. Int. J. Prod. Econ. *211*, 22–33.
39. Liu, Z., Hara, R., and Kita, H. (2021). Hybrid forecasting system based on data area division and deep learning neural network for short-term wind speed forecasting. Energy Convers. Manag. *238*, 114136.
40. Sinvaldo, R., Viviana, C., and Leandro, D. (2021). Hybrid multi-stage decomposition with parametric model applied to wind speed forecasting in Brazilian Northeast. Renew. Energy *164*, 1508–1526.
41. Duan, J., Wang, P., Ma, W., Fang, S., and Hou, Z. (2022). A novel hybrid model based on nonlinear weighted combination for short-term wind power forecasting. Int. J. Electr. Power Energy Syst. *134*, 107452.
42. Deng, S., Chen, Z., Kuang, F., Yang, C., and Gui, W. (2021). Optimal control of chilled water system with ensemble learning and cloud edge terminal implementation. IEEE Trans. Industr. Inform. *17*, 7839–7848.
43. Zhou, Z., Shojafar, M., Abawajy, J., Yin, H., and Lu, H. (2022). ECMS: an edge intelligent energy efficient model in mobile edge computing. IEEE Trans. Green Commun. Netw. *6*, 238–247.
44. Hou, W., Wen, H., Zhang, N., Lei, W., and Lin, H. (2022). Dynamic load combined prediction framework with collaborative cloud-edge for microgrid. In IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops, pp. 1–2.
45. Zhong, J., Liu, B., Yu, X., Wong, P., Wang, Z., Xu, C., and Zhou, X. (2023). Enhancing voltage compliance in distribution network under cloud and edge computing framework. IEEE Trans. Cloud Comput. *11*, 1217–1229. https://doi.org/10.1109/TCC.2022.3149238.
46. Li, J., Ren, Y., Fang, S., Li, K., and Sun, M. (2020). Federated learning-based ultra-short term load forecasting in power internet of things. In 2020 IEEE International Conference on Energy Internet, pp. 63–68.
47. Luo, A., Yuan, J., Liang, F., Yang, Q., and Mu, D. (2020). Load forecasting of electric vehicle charging station based on edge computing. In 2020 IEEE 3rd International Conference on Computer and Communication Engineering Technology, pp. 34–38.
48. Han, T., Muhammad, K., Hussain, T., Lloret, J., and Baik, S.W. (2021). An efficient deep learning framework for intelligent energy management in IoT networks. IEEE Internet Things J. *8*, 3170–3179.
49. Deb, P.K., Mondal, A., and Misra, S. (2022). AuGrid: edge-enabled distributed load management for smart grid service providers. IEEE Trans. Green Commun. Netw. *6*, 437–446.
50. Shi, J., Liu, N., Huang, Y., and Ma, L. (2022). An edge computing-oriented net power forecasting for PV-assisted charging station: model complexity and forecasting accuracy trade-off. Appl. Energy *310*, 118456.
51. Fujimoto, Y., Fujita, M., and Hayashi, Y. (2021). Deep reservoir architecture for short-term residential load forecasting: an online learning scheme for edge computing. Appl. Energy *298*, 117176.
52. Ma, Q., Shen, L., and Cottrell, G.W. (2020). DeePr-ESN: a deep projection-encoding echo-state network. Inf. Sci. *511*, 152–171.
53. Xi, W., Li, X., Feng, Q., Yao, H., Cai, T., and Yu, Y. (2022). Cyber security protection of power system equipment based on chip-level trusted computing. Front. Energy Res. *10*, 842938.
54. Wang, S., Wang, X., Wang, S., and Wang, D. (2019). Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting. Int. J. Electr. Power Energy Syst. *109*, 470–479.
55. Xu, X., and Meng, Z. (2020). A hybrid transfer learning model for short-term electric load forecasting. Electr. Eng. *102*, 1371–1381.
56. Wang, L., Lv, S.X., and Zeng, Y.R. (2018). Effective sparse AdaBoost method with ESN and FOA for industrial electricity consumption forecasting in China. Energy *155*, 1013–1031.
57. Huang, G., Wang, D., and Lan, Y. (2011). Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern. *2*, 107–122.
58. Xu, M., and Han, M. (2016). Adaptive elastic echo state network for multivariate time series prediction. IEEE Trans. Cybern. *46*, 2173–2183.
59. Huang, B., Qin, G., Zhao, R., Wu, Q., and Shahriari, A. (2018). Recursive Bayesian echo state network with an adaptive inflation factor for temperature prediction. Neural Comput. Appl. *29*, 1535–1543.
60. Dash, P., Rekha Pattnaik, S., NVDV Prasad, E., and Bisoi, R. (2023). Detection and classification of DC and feeder faults in DC microgrid using new morphological operators with multi class AdaBoost algorithm. Appl. Energy *340*, 121013.
61. Li, P., Ji, H., Wang, C., Zhao, J., Song, G., Ding, F., and Wu, J. (2019). Optimal operation of soft open points in active distribution networks under three-phase unbalanced conditions. IEEE Trans. Smart Grid *10*, 380–391.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
New South Wales (NSW) load dataset	Data.NSW	https://doi.org/10.1016/j.ijepes.2019.02.022
Global Energy Forecasting Competition 2012 load dataset	GEFCom2012	https://doi.org/10.1016/j.ijforecast.2013.07.001
Australian DKASC PV dataset	DKASC	https://dkasolarcentre.com.au/
Data used for simulation in this paper	Wang et al. (2024)	https://doi.org/10.5281/zenodo.11532169
Software and algorithms		
Matlab 2018	MathWorks	https://matlab.mathworks.com/
Python 3.8	Python	https://www.python.org/
Gurobi optimization toolbox	Gurobi	https://www.gurobi.com/
Original code in this paper	Wang et al. (2024)	https://doi.org/10.5281/zenodo.11532169

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Kangsheng Wang (wangkangsheng@tju.edu.cn).

Materials availability

This study did not generate new materials.

Data and code availability

- This paper analyzes existing, publicly available data which are listed in the key resources table. Accession numbers are listed in the key resources table. The data used for simulation in this paper from the publicly available datasets have been deposited at Zenodo (Wang et al., 2024) and are publicly available as of the date of publication. DOI is listed in the key resources table.
- All original code has been deposited at Zenodo (Wang et al., 2024) and is publicly available as of the date of publication. DOI is listed in the key resources table.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

METHOD DETAILS

Sparse integrated forecasting model under TrAdaBoost framework

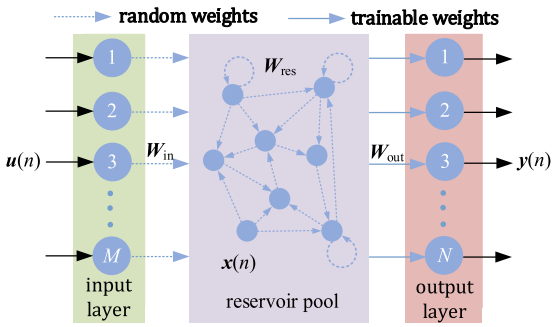
Echo state network

As a new type of recurrent neural network, ESN uses least squares estimation to calculate the output weight matrix, greatly improving the training efficiency. In addition, the main hyperparameters of ESN are only leaking parameter r, reservoir pool size L and regularization parameter C, so hyperparameter optimization occupies low computational resources, which is suitable as the core algorithm for edge-side load forecasting.

A typical structure of an echo state network is shown in the following figure, consisting of an input layer, a reservoir pool, and an output layer. Assume that the input vector dimension, reservoir pool size, and output vector dimension are M, L, and N, respectively. The transition of the echo state vector in the reservoir pool is defined as follows:

$$\boldsymbol{x}(n) = (1-r)\boldsymbol{x}(n-1) + r\cdot\tanh[\boldsymbol{W}_{\text{in}}\boldsymbol{u}(n) + \boldsymbol{W}_{\text{res}}\boldsymbol{x}(n-1)] \quad \text{(Equation 3)}$$

where r denotes the leaking parameter; tanh() denotes the hyperbolic tangent function; $\boldsymbol{x}(n) = [x_1(n), x_2(n), \ldots, x_L(n)]^{\mathrm{T}}$ is the echo state vector for the time step n; $\boldsymbol{x}(n-1)$ is the echo state vector for the time step $n-1$; $\boldsymbol{u}(n)$ is the input vector for the time step n; $\boldsymbol{W}_{\text{in}} \in \mathbb{R}^{L\times M}$ and $\boldsymbol{W}_{\text{res}} \in \mathbb{R}^{L\times L}$ are the weights connecting the input layer to the reservoir pool and the weights connecting the neurons in the reservoir pool, respectively; elements of $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}_{\text{res}}$ are generated by random numbers uniformly distributed on $[-1, 1]$, and $\boldsymbol{W}_{\text{res}}$ additionally satisfies its spectral radius less than 1 to ensure that the reservoir pool has rich nonlinearity.



Architecture of the echo state network

ESN obtains the forecasting result $\boldsymbol{y}(n)$ by linear regression of the input vector $\boldsymbol{u}(n)$ and the echo state vector $\boldsymbol{x}(n)$:

$$\boldsymbol{y}(n) = \boldsymbol{W}_{\text{out}}[\boldsymbol{u}(n); \boldsymbol{x}(n)] \quad \text{(Equation 4)}$$

where $\boldsymbol{W}_{\text{out}}$ is the weights to be solved connecting the reservoir pool and the output layer.

Assume that there are $n_0 + \Gamma$ training samples $\{(\boldsymbol{u}(1), \boldsymbol{y}_d(1)), (\boldsymbol{u}(2), \boldsymbol{y}_d(2)), \ldots, (\boldsymbol{u}(n_0+\Gamma), \boldsymbol{y}_d(n_0+\Gamma))\}$ in the training set. Each sample is loaded using Equation 1 and the echo state vector at each time step are stored. To solve the initial value instability problem, the first n_0 samples are discarded. Let $\boldsymbol{s}(n) = [\boldsymbol{u}(n); \boldsymbol{x}(n)]$, and then generate the reservoir pool state matrix $\boldsymbol{S} \in \mathbb{R}^{(M+L)\times\Gamma}$ and target output matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times\Gamma}$:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{s}_1(n_0+1) & \cdots & \boldsymbol{s}_1(n_0+\Gamma) \\ \vdots & \ddots & \vdots \\ \boldsymbol{s}_{M+L}(n_0+1) & \cdots & \boldsymbol{s}_{M+L}(n_0+\Gamma) \end{bmatrix} \quad \text{(Equation 5)}$$

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1(n_0+1) & \cdots & \boldsymbol{y}_1(n_0+\Gamma) \\ \vdots & \ddots & \vdots \\ \boldsymbol{y}_N(n_0+1) & \cdots & \boldsymbol{y}_N(n_0+\Gamma) \end{bmatrix} \quad \text{(Equation 6)}$$

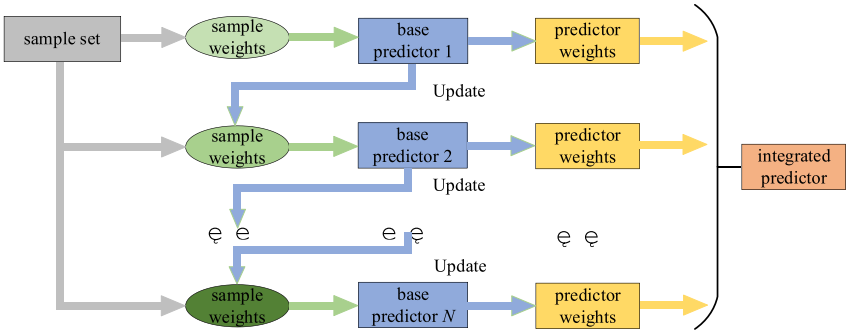
To prevent pathological solutions and to improve the generalization ability of ESN, ridge regression is generally used to solve for $\boldsymbol{W}_{\text{out}}$:

$$\boldsymbol{W}_{\text{out}} = \boldsymbol{Y}\boldsymbol{S}^{\mathsf{T}}\left(\boldsymbol{S}\boldsymbol{S}^{\mathsf{T}} + \frac{1}{C}\boldsymbol{I}\right)^{-1} \quad \text{(Equation 7)}$$

where C is regularization parameter; $\boldsymbol{I}$ is unit matrix.

TrAdaBoost framework

ESN has few hyperparameters and a simple solution process, so it is chosen as the core algorithm for edge-side load forecasting in this study. However, since $\boldsymbol{W}_{\text{in}}$ and $\boldsymbol{W}_{\text{res}}$ are randomly generated and the one-time training mechanism, the forecasting performance of a single ESN is not stable in practice. Influenced by the external environment, the distribution characteristics of edge-side load and its influencing factors change significantly over time. In addition, the number of training set samples is small compared to hourly forecasting, so training with only temporally recent samples faces the difficulty of insufficient training samples. A typical ensemble learning framework with transfer learning style, TrAdaBoost, can be employed to effectively solve the above problems, and its integration process is shown in the following figure.



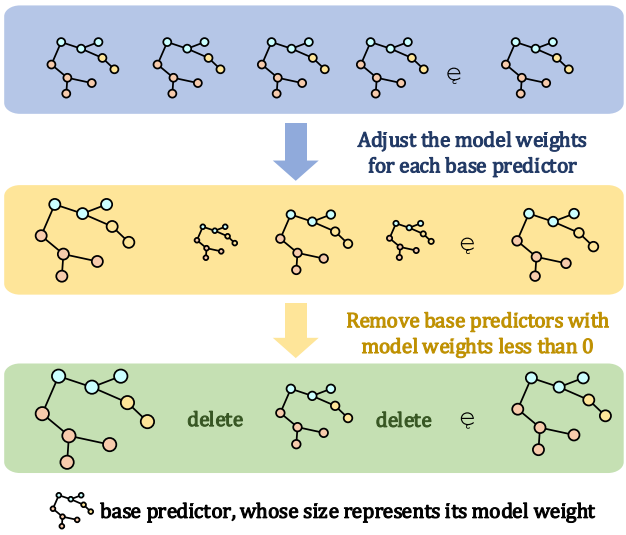
Integration process of TrAdaBoost

TrAdaBoost assigns an initial weight to each sample in the source domain and the target domain in the initial stage. If a sample in the target domain has a high loss, its weight is increased in the next iteration. If a sample in the source domain has a high loss, it is considered unfavorable to learn the target domain task, and therefore its weight will be reduced in the next iteration. By using different sample weight adjustment strategies for the source and target domain samples, the influence of the source domain samples can be reduced to help the target domain learn an efficient integrated model. Therefore, TrAdaBoost is selected as the ensemble learning framework in this paper.

Adaptive sparse integration method

Ensemble learning requires a large number of base predictors for integration, which has a huge computational overhead in forecasting. Meanwhile, the model parameters of each base predictor occupy significant storage resources, further limiting the practical application of ensemble learning model in edge computing.[56] Therefore, this study further improves the adaptive sparse integration method in 56, extending from single-step forecasting to multi-step forecasting, and modifies the searching process of key base predictors. The improved method supports multi-step and multi granularity prediction. Compared to retraining, the daily correction method has a faster calculation speed and can save approximately 99% of storage resources.

In each round of iteration, two key base predictors are searched from the added base predictors. By redistributing the optimal model weight p^* between these two predictors, the model weight of one base predictor is increased and the model weight of the other one is decreased. When the model weight of a base predictor is less than 0, the base predictor is removed, thus achieving a reduction in the final integrated model size. The adaptive sparse integration process is shown in the following figure.



Process of adaptive sparse integration

Assuming that the number of base ESN predictors is N_0, the training set Ω has a total of Γ samples, the input vector dimension is $M \times 1$, and the output label dimension is $N \times 1$. The training set Ω is divided into the source domain data set Ω_s (the first M_d samples) and the target domain data set Ω_t (the remaining N_d samples) in chronological order. The specific steps are as follows:

Step 1): Set source domain weight parameter $\beta = \frac{1}{1+\sqrt{\frac{2\ln M_d}{N_0}}}$, and current iteration number $l = 1$. Calculate initial weights $\boldsymbol{D}^0 = \{w_i^0\}$ for each sample of the training set Ω:

$$w_i^0 = \begin{cases} \frac{1}{2M_d}, i = 1, 2, \ldots, M_d \\ \frac{1}{2N_d}, i = M_d + 1, \ldots, M_d + N_d \end{cases} \quad \text{(Equation 8)}$$

$A = \pi r^2$Step 2): For the training set Ω with a weight distribution $\boldsymbol{D}^{l-1}$, calculate maximum regression error E_l for each sample:

$$E_l = \max\left(\sum |\boldsymbol{y}_d(i) - \boldsymbol{G}_l(\boldsymbol{u}(i))|\right), i = 1, 2, \ldots, M_d + N_d \quad \text{(Equation 9)}$$

where $\boldsymbol{G}_l(\boldsymbol{u}(i))$ denotes the predicted value of $\boldsymbol{G}_l(\boldsymbol{x})$ for the i th sample, $\boldsymbol{G}_l(\boldsymbol{x})$ denotes the l th base ESN predictor; $\boldsymbol{y}_d(i)$ is the actual value of the i th sample with dimension $N \times 1$; $\sum |\boldsymbol{y}|$ denotes the sum of absolute values of all elements of $\boldsymbol{y}$.

Step 3): Calculate relative regression error $e_{l,i}$ for $\boldsymbol{G}_l(\boldsymbol{x})$ on the training set Ω:

$$e_{l,i} = \frac{\sum|\boldsymbol{y}_{\mathrm{d}}(i) - \boldsymbol{G}_l(\boldsymbol{u}(i))|}{E_l}, i = 1, 2, \ldots, M_{\mathrm{d}}+N_{\mathrm{d}} \quad \text{(Equation 10)}$$

Step 4): Calculate overall regression error e_l and β_l for $\boldsymbol{G}_l(\boldsymbol{x})$ on the training set Ω:

$$e_1 = min\left(\sum_{i=1}^{M_d+N_d} w_i^{l=1} e_{l,i}, 0.5\right) \quad \text{(Equation 11)}$$

$$\beta_l = \frac{e_l}{1 - e_l} \quad \text{(Equation 12)}$$

where $min(\sum_{i=1}^{M_d+N_d} w_i^{l=1} e_{l,i}, 0.5)$ denotes the smaller value of $\sum_{i=1}^{M_d+N_d} w_i^{l-1} e_{l,i}$ and 0.5.

Step 5): Calculate model weight coefficient α_l for $\boldsymbol{G}_l(\boldsymbol{x})$ and cumulative model weight coefficient R:

$$\alpha_l = \ln\frac{1}{\beta_l} \quad \text{(Equation 13)}$$

$$R = \sum_{i=1}^{l} \alpha_i \quad \text{(Equation 14)}$$

Step 6): Find two key base ESN predictor indexes for model weight adjustment, with subscripts noted as h_1 and h_2 (if $l = 1$, then $l = l + 1$, return to Step 2)):

$$h_1 = \underset{h\in[1,l]}{\operatorname{argmin}} \sum_{i=M_{\mathrm{d}}+1}^{M_{\mathrm{d}}+N_{\mathrm{d}}} \sum \boldsymbol{G}_h(\boldsymbol{u}(i)) \circ \left[\boldsymbol{y}_{\mathrm{d}}(i) - \sum_{c=1}^{l} \frac{\alpha_c}{R} \boldsymbol{G}_c(\boldsymbol{u}(i))\right] \quad \text{(Equation 15)}$$

$$h_2 = \underset{h\in[1,l]}{\operatorname{argmax}} \sum_{i=M_{\mathrm{d}}+1}^{M_{\mathrm{d}}+N_{\mathrm{d}}} \sum \boldsymbol{G}_h(\boldsymbol{u}(i)) \circ \left[\boldsymbol{y}_{\mathrm{d}}(i) - \sum_{c=1}^{l} \frac{\alpha_c}{R} \boldsymbol{G}_c(\boldsymbol{u}(i))\right] \quad \text{(Equation 16)}$$

where '$\circ$' denotes Hadamard product (corresponding elements of matrix are multiplied).

Step 7): Calculate optimal model weight moving step p^* and adjust model weights of the selected base ESN predictor:

$$p^* = \frac{R \sum_{i=M_{\mathrm{d}}+1}^{M_{\mathrm{d}}+N_{\mathrm{d}}} \sum\left[\boldsymbol{y}_{\mathrm{d}}(i) - \sum_{c=1}^{l} \frac{\alpha_c}{R} \boldsymbol{G}_c(\boldsymbol{u}(i))\right] \circ \left[\boldsymbol{G}_{h_2}(\boldsymbol{u}(i)) - \boldsymbol{G}_{h_1}(\boldsymbol{u}(i))\right]}{\sum_{i=M_{\mathrm{d}}+1}^{M_{\mathrm{d}}+N_{\mathrm{d}}} \sum\left[\boldsymbol{G}_{h_2}(\boldsymbol{u}(i)) - \boldsymbol{G}_{h_1}(\boldsymbol{u}(i))\right] \circ \left[\boldsymbol{G}_{h_2}(\boldsymbol{u}(i)) - \boldsymbol{G}_{h_1}(\boldsymbol{u}(i))\right]} \quad \text{(Equation 17)}$$

$$\begin{cases} \alpha_{h_1} = \alpha_{h_1} - p^* \\ \alpha_{h_2} = \alpha_{h_2} + p^* \end{cases} \quad \text{(Equation 18)}$$

Step 8): If $\min(\alpha_{h_1}, \alpha_{h_2}) < 0$, adjust cumulative model weight coefficients R and remove the base ESN predictor with model weights less than 0:

$$R = R - \min\left(\alpha_{h_1}, \alpha_{h_2}\right) \quad \text{(Equation 19)}$$

Step 9): Update the sample weights of the source and target domains:

$$w_i^l = \begin{cases} \dfrac{w_i^{l-1}\beta^{e_{l,i}}}{\sum_{i=1}^{M_{\mathrm{d}}} w_i^{l-1}\beta^{e_{l,i}}}, i = 1, 2, \ldots, M_{\mathrm{d}} \\ \dfrac{w_i^{l-1}\beta_t^{-e_{l,i}}}{\sum_{i=M_{\mathrm{d}}+1}^{M_{\mathrm{d}}+N_{\mathrm{d}}} w_i^{l-1}\beta_t^{-e_{l,i}}}, i = M_{\mathrm{d}}+1, \ldots, M_{\mathrm{d}}+N_{\mathrm{d}} \end{cases} \quad \text{(Equation 20)}$$

Step 10): If $l < N_0$, then set $l = l + 1$ and return to Step 2); otherwise obtain initial integrated ESN forecasting model $\boldsymbol{G}(\boldsymbol{x})$:

$$G(\boldsymbol{x}) = \sum_{i=1}^{N_0} \frac{\alpha_i}{\sum_{l=1}^{N_0} \alpha_l} G_i(\boldsymbol{x}) \quad \text{(Equation 21)}$$

Auto-encoder for model dimensionality reduction

ESN uses high-dimensional projections to capture dynamic changes in sequences, and the reservoir pool is typically composed of sparsely connected neurons of 100-1000 times the input dimension 52. The high sparsity leads to the fact that when performing online forecasting, all echo state vectors are involved in the computation, many of which are high frequency components that have little impact on the output. To further reduce computational and storage overhead during online forecasting, it is necessary to project original high-dimensional features into low-dimensional feature space and extract principal components.

The main widely used methods for feature dimensionality reduction are principal component analysis and auto-encoder (AE), in which the auto-encoder tends to have lower complexity. A simplified auto-encoder for ELM is proposed in.[57] Since output weights of both ELM and ESN are obtained by linear regression, the simplified auto-encoder in 57 is also applicable to ESN. With this method, the high-dimensional state matrix $\boldsymbol{S} \in \mathbb{R}^{(M+L)\times T}$ is mapped to low-dimensional state matrix $\boldsymbol{H} \in \mathbb{R}^{D_L \times T}$ through random weight matrix $\boldsymbol{A} \in \mathbb{R}^{D_L \times (M+L)}$ and random bias matrix $\boldsymbol{B} \in \mathbb{R}^{D_L \times T}$, where D_L is the setting low-dimensional dimension:

$$\boldsymbol{H} = \boldsymbol{A}\boldsymbol{S} + \boldsymbol{B} \quad \text{(Equation 22)}$$

where each element of $\boldsymbol{A}$ and $\boldsymbol{B}$ is generated by a random number uniformly distributed on [-1,1].

The encoding weight matrix $\boldsymbol{W}_{enc} \in \mathbb{R}^{D_L \times (M+L)}$ is determined by the following optimization problem:

$$\boldsymbol{W}_{enc} = \underset{W}{\operatorname{argmin}} \|\boldsymbol{W}\boldsymbol{S} - \boldsymbol{H}\|_2 + \frac{1}{C}\|\boldsymbol{W}\|_2 \quad \text{(Equation 23)}$$

$\boldsymbol{W}_{enc}$ can be directly solved by ridge regression:

$$\boldsymbol{W}_{enc} = \boldsymbol{H}\boldsymbol{S}^{\mathrm{T}}\left(\boldsymbol{S}\boldsymbol{S}^{\mathrm{T}} + \frac{1}{C}\boldsymbol{I}\right)^{-1} \quad \text{(Equation 24)}$$

With the introduction of $\boldsymbol{W}_{enc}$, $\boldsymbol{W}_{out}$ represents the mapping relationship from low-dimensional state vector $\boldsymbol{h}(n)$ to target output $\boldsymbol{y}(n)$:

$$\boldsymbol{W}_{out} = \boldsymbol{Y}\boldsymbol{H}^{\mathrm{T}}\left(\boldsymbol{H}\boldsymbol{H}^{\mathrm{T}} + \frac{1}{C}\boldsymbol{I}\right)^{-1} \quad \text{(Equation 25)}$$

Accordingly, the steps in forecasting are as follows:

$$\begin{cases} \boldsymbol{x}(n) = (1-r)\boldsymbol{x}(n-1) + r\cdot\tanh[\boldsymbol{W}_{in}\boldsymbol{u}(n) + \boldsymbol{W}_{res}\boldsymbol{x}(n-1)] \\ \boldsymbol{h}(n) = \boldsymbol{W}_{enc}[\boldsymbol{u}(n); \boldsymbol{x}(n)] \\ \boldsymbol{y}(n) = \boldsymbol{W}_{out}\boldsymbol{h}(n) \end{cases} \quad \text{(Equation 26)}$$

Adaptive correction of the forecasting model

Daily update of the forecasting model

As a widely used data assimilation method, Kalman filtering recursively obtains the optimal estimate of current state of system based on state space equations and minimum mean square error estimation criterion. Ref.[58] proposed state space equation of ESN by considering $\boldsymbol{W}_{out}$ as state variables. It is worth noting that the dimension of $\boldsymbol{W}_{out}$ is $N \times L$ when performing the day-ahead forecasting. Thus, $\boldsymbol{W}_{out}$ are decoupled into N independent state variables by forecasting time interval and corrected separately.

Assume that after sparse integration, $\boldsymbol{G}(\boldsymbol{x})$ consists of N_{sp} base ESN predictors $\{\boldsymbol{G}_i(\boldsymbol{x})\}$; the i-th base predictor on day n is $\boldsymbol{G}_{n,i}(\boldsymbol{x})$, its output weight matrix is $\boldsymbol{W}_{out}^{n,i}$, its encoding weight matrix is $\boldsymbol{W}_{enc}^{i}$, its model weight is α_i, and the corresponding hyperparameter set is $\{r_i, L_i, C_i\}$. The state space equations of $\boldsymbol{G}_{n,i}(\boldsymbol{x})$ are specified as follows:

$$\boldsymbol{W}_{out}^{n,i}(m) = \left[\boldsymbol{W}_{out}^{n,i}(m,:)\right]^{\mathrm{T}}, m = 1, 2, \ldots, N \quad \text{(Equation 27)}$$

$$\boldsymbol{W}_{out}^{n,i}(m) = \boldsymbol{W}_{out}^{n-1,i}(m) + \boldsymbol{\epsilon}(n-1), m = 1, 2, \ldots, N \quad \text{(Equation 28)}$$

$$y_d^n(m) = \boldsymbol{h}^{\mathrm{T}}(n)\boldsymbol{W}_{out}^{n,i}(m) + \sigma(m), m = 1, 2, \ldots, N \quad \text{(Equation 29)}$$

where $\boldsymbol{W}_{out}^{n,i}(m,:)$ denotes the m th row of $\boldsymbol{W}_{out}^{n,i}$; $y_d^n(m)$ denotes the actual value of the m th time interval on the n th day; $\boldsymbol{\epsilon}(n-1)$ denotes systematic noise of state variables; $\boldsymbol{Q}_{n-1}(m)$ denotes covariance of $\boldsymbol{\epsilon}(n-1)$; and $\sigma(m)$ denotes measurement noise of the m th time interval.

The update step of $\boldsymbol{W}_{out}^{n,i}$ consists of two parts: time update and measurement update, in which the time update is as follows:

$$\boldsymbol{P}_{n,i}^{-}(m) = \boldsymbol{P}_{n-1,i}(m) + \boldsymbol{Q}_{n-1,i}(m), m = 1, 2, \ldots, N \quad \text{(Equation 30)}$$

where $\boldsymbol{P}_{n-1,i}(m)$ denotes posterior estimation error covariance of $\boldsymbol{W}_{out}^{n-1,i}(m)$; $\boldsymbol{Q}_{n-1,i}(m)$ denotes the system noise covariance; and $\boldsymbol{P}_{n,i}^{-}(m)$ denotes the prior estimation error covariance of $\boldsymbol{W}_{out}^{n,i}(m)$.

Measurement update consists of calculating Kalman gain $K_{n,i}(m)$ and updating the covariance matrix:

$$K_{n,i}(m) = P^{-}_{n,i}(m)h_i(n)\left[h_i^{\mathrm{T}}(n)P^{-}_{n,i}(m)h_i(n)+\sigma(m)\right]^{-1}, m = 1,2,\ldots,N \quad \text{(Equation 31)}$$

$$P_{n,i}(m) = \left[I - K_{n,i}(m)h^{\mathrm{T}}(n)\right]P^{-}_{n,i}(m), m = 1,2,\ldots,N \quad \text{(Equation 32)}$$

$$W^{n+1,i}_{\mathrm{out}}(m) = W^{n,i}_{\mathrm{out}}(m) + K_{n,i}(m)\left[y^{n}_{\mathrm{d}}(m) - h^{\mathrm{T}}(n)W^{n,i}_{\mathrm{out}}(m)\right], m = 1,2,\ldots,N \quad \text{(Equation 33)}$$

$$W^{n+1,i}_{\mathrm{out}} = \left[W^{n+1,i^{\mathrm{T}}}_{\mathrm{out}}(1); W^{n+1,i^{\mathrm{T}}}_{\mathrm{out}}(2); \ldots; W^{n+1,i^{\mathrm{T}}}_{\mathrm{out}}(N)\right] \quad \text{(Equation 34)}$$

However, in practical applications, it is difficult to determine system noise as well as measurement noise of load. For this problem, ref.[59] suggests to correct covariance using measurement data during measurement update. In this study, the covariance is corrected every k days:

$$F_{\mathrm{e}}(m) = \frac{c(m) - \sigma(m)}{h_i^{\mathrm{T}}(n)P^{-}_{n,i}(m)h_i(n)}, m = 1,2,\ldots,N \quad \text{(Equation 35)}$$

$$P_{n,i}(m) = F_{\mathrm{e}}(m)\left(I - K_{n,i}(m)h_i^{\mathrm{T}}(n)\right)P^{-}_{n,i}(m), m = 1,2,\ldots,N \quad \text{(Equation 36)}$$

$$c(m) = \sum_{j=0}^{k}\frac{1}{k+1}\left(y^{n-j}_{\mathrm{d}}(m) - y^{n-j}(m)\right)^2, m = 1,2,\ldots,N \quad \text{(Equation 37)}$$

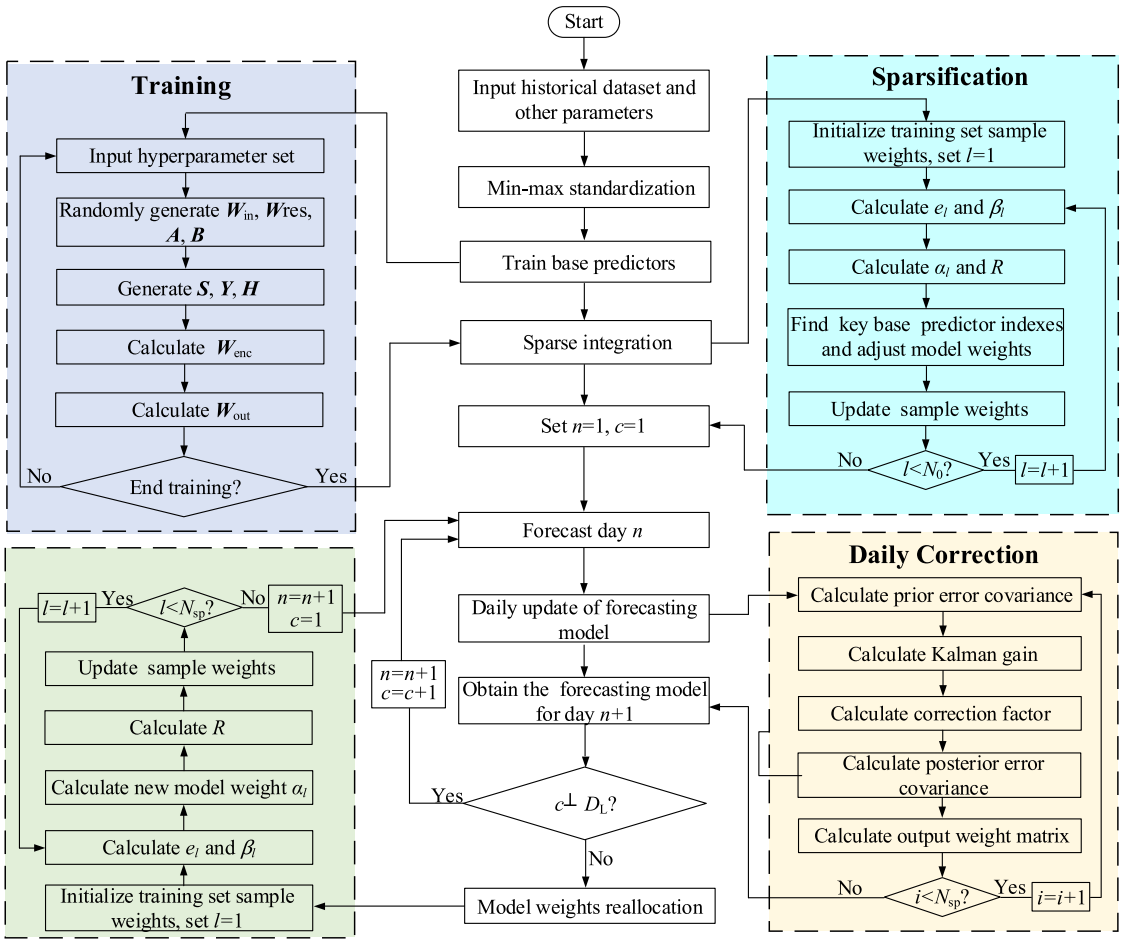
where $F_{\mathrm{e}}(m)$ is correction factor for the m th time interval; $c(m)$ is variance of forecasting error at the m th time interval from day $n - k$ to day n; and $y^{n-j}_{\mathrm{d}}(m)$ and $y^{n-j}(m)$ denote actual load value and forecasting value, respectively; k is sample size of correction factor.

Model weight reallocation in long-term forecasting

The daily update mechanism allows each base predictor to continuously learn the gradual changes in load, ensuring short-term adaptability. However, the base predictor with the best forecasting performance in the previous period does not guarantee that it will remain optimal in the future. Therefore, reallocating the model weight coefficient of each base predictor at regular intervals becomes necessary to address the long-term adaptivity problem. In this study, model weights $\{\alpha_i\}$ of base predictor $\{G_i(x)\}$ are reassigned using AdaBoost algorithm[60] every T_{L} days to obtain new model weights $\{\alpha'_i\}$, where T_{L} is the long-term forecasting days threshold.

Overall procedure of the proposed forecasting method

As shown in the following figure, the overall process of the proposed adaptive integrated load forecasting method for edge computing is as follows:



Flow chart of the proposed method

Step 1): Input load history data as the initial training set. Input n_0, Γ, leaking parameter candidate set $\{r_1, r_2, \ldots, r_{N_1}\}$, reservoir pool size candidate set $\{L_1, L_2, \ldots, L_{N_2}\}$, regularization parameter candidate set $\{C_1, C_2, \ldots, C_{N_3}\}$, $N_0 = N_1 \times N_2 \times N_3$, D_L, M_d, N_d, k, T_L and other parameters;

Step 2): Perform min-max normalization and sample partitioning to obtain the new training set Ω';

Step 3): Train N_0 base ESN predictors based on the training set 'Ω;

Step 4): Perform sparse integration to obtain the initial model $\boldsymbol{G}(\boldsymbol{x})$ as the forecasting model on day 1. Set $n = 1$, $c = 1$;

Step 5): Forecast the load on day n with the model on day n, and obtain forecasting result $\boldsymbol{y}_{\text{fore}}^{n}$;

Step 6): At the end of day t, record actual load data $\boldsymbol{y}_{\text{d}}^{n}$ and update the forecasting model on day n. After updating parameters, the new model is used for the forecasting on day $n+1$;

Step 7): If $c \leq T_L$, set $n = n+1$, $c = c+1$, then return to step 5); otherwise proceed to step 8);

Step 8): Reassign model weights of the base predictor, set $n = n+1$, $c = 1$, then return to step 5) to complete the following forecasting tasks.

Algorithm complexity and storage overhead analysis

Model training complexity analysis. The model complexity determines the computation time and the feasibility of deployment on edge-computing devices. The computational overhead of considering auto-encoder mainly lies in the addition of Equations 20 and 22, whose complexity is $\mathbf{O}[\max(\Delta_L^3, D_L^2\Gamma, D_L\Gamma \times (M+L))]$. Since $D_L \ll M+L$, after introducing auto-encoder, computational overhead increases very little during the initial training.

Offline and online forecasting complexity comparison. Offline prediction refers to make a prediction by training the model based on a training set, which needs to retrain the model periodically. Online forecasting refers to make a prediction by correcting the model with new measured data.

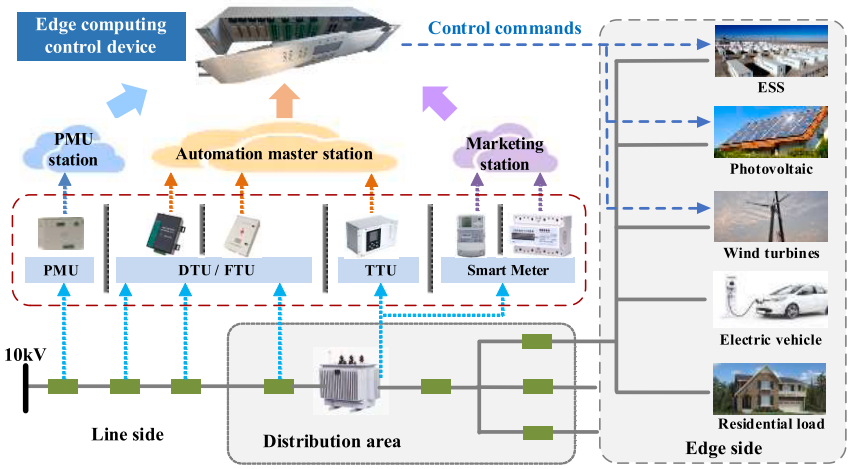
The computational overhead of offline forecasting is mainly in the inverse of the high-dimensional matrix $\left(\boldsymbol{S}\boldsymbol{S}^{\mathrm{T}}+\frac{1}{C}\boldsymbol{I}\right)$, and its complexity is $\mathrm{O}\left((\mathrm{M}+\Lambda)^3\right)$, which needs to calculate N_0 times. The main computational overhead of online forecasting lies in computing Kalman gain $\boldsymbol{K}$, whose complexity is $\mathrm{O}((\mathrm{M}+\Lambda)^2)$, requiring to calculate $N_{sp} \times N$ times. Since $M+L$ is in the order of 10^3, $\mathrm{O}((\mathrm{M}+\Lambda)^2) \ll \mathrm{O}((\mathrm{M}+\Lambda)^3)$, so computational overhead of online forecasting is much lower.

Complexity and storage overhead of online forecasting. Computational overhead of online forecasting is lower, but accordingly, online forecasting requires additional storage of parameters. Without auto-encoder, Kalman gain $\boldsymbol{K}$ must be calculated based on the high-dimensional vector $\boldsymbol{s}$ with complexity of $\mathrm{O}((\mathrm{M}+\Lambda)^2)$. With the employment of auto-encoder, computational complexity is $\mathrm{O}(\Delta_{\mathrm{L}}^2)$ based on the low-dimensional vector $\boldsymbol{h}$. In addition, in terms of storage resource consumption, the model without auto-entcoder needs to store $N_{sp} \times N$ covariance matrices $\{\boldsymbol{P}\}$ with dimensionality $(M+L) \times (M+L)$. With auto-encoder, the number of $\{\boldsymbol{P}\}$ remains the same, but the dimensionality is only $D_{\mathrm{L}} \times D_{\mathrm{L}}$. Since $D_{\mathrm{L}} \ll M+L$, computational complexity and storage overhead of the model with auto-encoder are greatly reduced, which is more suitable for edge computing.

MULTI-TIMESCALE LOCAL PREDICTIVE CONTROL FOR EDGE COMPUTING

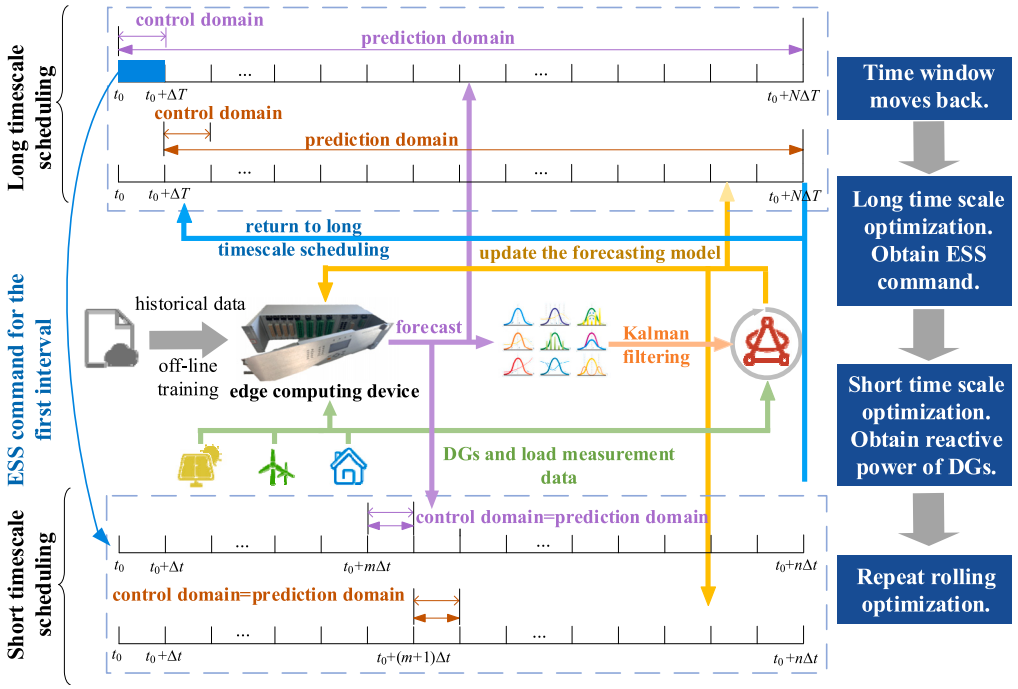
Multi-timescale model predictive control architecture

As shown in the following figure, a typical edge-computing application scenario in active distribution network with local loads, DGs, flexible power distribution devices, and energy storage system (ESS) is considered. The charging and discharging strategy of ESS needs to be optimized as a whole with multi-timescale forecasting information. The proposed forecasting method combines accuracy and complexity, while being able to support different time scales and quickly correct the model. So, model predictive control is chosen to minimize the total system network loss for each dispatch day. By continuously updating DGs and load forecasting information in the remaining time periods, long time scale economic operation and short time scale safe operation at the edge side are realized.



A typical edge-computing application scenario

The principle of MPC-based rolling optimization is shown in the following figure. We take the whole day as a complete scheduling cycle. The forecasting window of first long time-scale dispatching is from 0:00 to 24:00. During the long time-scale dispatching period, at the moment t_0, based on the forecasting information of the prediction domain (from t_0 to $t_0+N\Delta T$, where ΔT is execution interval of long time-scale optimization), the strategy of ESS and reactive power command in the prediction domain are optimally obtained, but only the strategy of ESS in the first interval is executed. In the next scheduling period, the prediction domain moves forward and updates the forecasting information of the remaining period. The rolling optimization is carried out until the whole scheduling cycle is completed.



MPC-based multi-timescale rolling optimization

DGs and load are strongly influenced by environmental factors, and the minute level output may deviate seriously from the hourly average output, thus requiring dynamic optimization on a smaller time scale. Considering the limited computing resources of edge-computing devices, it is difficult to solve large optimization problems within a few minutes. Therefore, ΔT is further divided into multiple short time-scale execution interval Δt. During the short-time optimization period (the first interval of the latest long time-scale dispatching period), the strategy of ESS obtained from long-time optimization remains unchanged, and the optimization variables are only reactive power output of DGs.

Edge-side operation optimization model

In this paper, the long sliding window ΔT is 1 hour and the short sliding window Δt is 15 minutes.

Long-time operation objective function. The long-time operation mainly takes into account the economy and safety, with total network loss and total voltage deviation as the optimization objectives. We weight coefficients λ_1 and λ_2 to flexibly balance safety and economy based on actual operational requirements.

$$\min f_a = \lambda_1 f_{L,a} + \lambda_2 f_{V,a} \quad \text{(Equation 38a)}$$

$$f_{L,a} = \sum_{t=1}^{N_T} \sum_{ij \in L_a} R_{ij} I_{t,ij} \quad \text{(Equation 38b)}$$

$$f_{V,a} = \sum_{t=1}^{N_T} \sum_{i=1}^{N_B} A_{t,i} \quad \text{(Equation 38c)}$$

$$A_{t,i} \geq v_{t,i} - \left(V_{thr}^{max}\right)^2 \quad \text{(Equation 38d)}$$

$$A_{t,i} \geq -v_{t,i} + \left(V_{thr}^{min}\right)^2 \quad \text{(Equation 38e)}$$

$$A_{t,i} \geq 0 \quad \text{(Equation 38f)}$$

where $f_{L,a}$ is the total system network loss; $f_{V,a}$ is the total system voltage deviation; λ_1 and λ_2 are weight coefficients for $f_{L,a}$ and $f_{V,a}$; N_T is the number of remaining time intervals; N_B is the number of nodes; $v_{t,i} = V_{t,i}^2$, $l_{t,ij} = I_{t,ij}^2$ are the replacement variables introduced by the cone transformation; $V_{t,i}$ denotes the voltage amplitude at node i at time interval t; $I_{t,ij}$ denotes the current amplitude flowing between nodes i and j at time interval t; R_{ij} denotes the resistance of branch ij; V_{thr}^{max} and V_{thr}^{min} denote the expected upper and lower voltage limits.

Short-time operation objective function:. The objective function is the total system voltage deviation and network loss the next scheduling period:

$$\min f_s = \lambda_1 \sum_{t=1}^{N_T} \sum\nolimits_{ij \in L_a} R_{ij} I_{t,ij} + \lambda_2 \sum_{t=1}^{N_T} \sum_{i=1}^{N_B} A_{t,i} \quad \text{(Equation 39)}$$

where A_i is the voltage deviation of node i.

Constraints. The constraints of the edge-side operational optimization model contain system power flow constraints, system operation security constraints, DG operation constraints, and ESS operation constraints. The details are as follows:

System power flow constraints

$$\sum\nolimits_{ji \in L_a} \left(P_{t,ji} - R_{ji} I_{t,ji}\right) + P_{t,i} = \sum\nolimits_{ik \in L_a} P_{t,ik} \quad \text{(Equation 40a)}$$

$$\sum\nolimits_{ji \in L_a} \left(Q_{t,ji} - X_{ji} I_{t,ji}\right) + Q_{t,i} = \sum\nolimits_{ik \in L_a} Q_{t,ik} \quad \text{(Equation 40b)}$$

$$v_{t,i} - v_{t,j} + \left(R_{ij}^2 + X_{ij}^2\right) I_{t,ij} = 2\left(R_{ij} P_{t,ij} + X_{ij} Q_{t,ij}\right) \quad \text{(Equation 40c)}$$

$$I_{t,ij} v_{t,i} = P_{t,ij}^2 + Q_{t,ij}^2, ij \in L_a \quad \text{(Equation 40d)}$$

$$P_{t,i} = P_{t,i}^{DG} - P_{t,i}^{L}, i \in \Omega_a \quad \text{(Equation 40e)}$$

$$Q_{t,i} = Q_{t,i}^{DG} - Q_{t,i}^{L}, i \in \Omega_a \quad \text{(Equation 40f)}$$

where L_a and Ω_a denote the set of branches and the set of nodes in the area covered by the edge-computing device, respectively; $P_{t,ji}$ and $Q_{t,ji}$ denote the active and reactive power flowing from node j to node i at time interval t, respectively; $P_{t,ik}$ and $Q_{t,ik}$ denote the active and reactive power flowing from node i to node k at time interval t, respectively; R_{ij} and X_{ij} denote the resistance and reactance of branch ij, respectively; $P_{t,i}$ and $Q_{t,i}$ denote the net active power and net reactive power at node i at time interval t, respectively; $P_{t,i}^{DG}$ and $Q_{t,i}^{DG}$ denote the active power and reactive power of DG at node i at time interval t, respectively; $P_{t,i}^{L}$ and $Q_{t,i}^{L}$ denote the active and reactive power of load at node i at time interval t.

System operation security constraints

$$(\underline{V})^2 \leq v_{t,i} \leq (\overline{V})^2, i \in \Omega_a \quad \text{(Equation 41a)}$$

$$I_{t,ij} \leq (\overline{I})^2, ij \in L_a \quad \text{(Equation 41b)}$$

where $\underline{V}$ and $\overline{V}$ are the minimum and maximum values of voltage for safe operation, respectively; $\overline{I}$ is the maximum value of branch circuit current for safe operation.

DG operation constraints

$$Q_{t,i}^{DG,max} = \sqrt{\left(S_i^{DG}\right)^2 - \left(P_{t,i}^{DG}\right)^2}, i \in \Omega_a \quad \text{(Equation 42a)}$$

$$-Q_{t,i}^{DG,max} \leq Q_{t,i}^{DG} \leq Q_{t,i}^{DG,max}, i \in \Omega_a \quad \text{(Equation 42b)}$$

where S_i^{DG} is the DG capacity at node i.

ESS operation constraints

$$\left(P_{t,i}^{ESS}\right)^2 + \left(Q_{t,i}^{ESS}\right)^2 \leq \left(S_i^{ESS}\right)^2, i \in \Omega_a \quad \text{(Equation 43a)}$$

$$-P_i^{ESS,max} \leq P_{t,i}^{ESS} \leq P_i^{ESS,max}, i \in \Omega_a \quad \text{(Equation 43b)}$$

$$-Q_i^{ESS,max} \leq Q_{t,i}^{ESS} \leq Q_i^{ESS,max}, i \in \Omega_a \quad \text{(Equation 43c)}$$

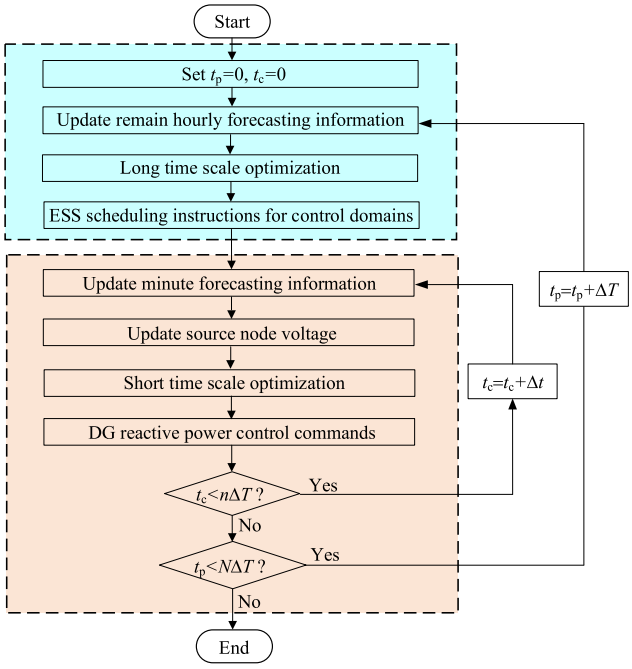
$$SOC_{t,i}^{ESS} = SOC_{t-1,i}^{ESS} + \frac{P_{t,i}^{ESS}}{S_i^{ESS}}, i \in \Omega_a \quad \text{(Equation 43d)}$$

$$SOC_i^{ESS,min} \leq SOC_{t,i}^{ESS} \leq SOC_i^{ESS,max}, i \in \Omega_a \quad \text{(Equation 43e)}$$

$$SOC_{0,i}^{ESS} = SOC_{N_T,i}^{ESS}, i \in \Omega_a \quad \text{(Equation 43f)}$$

where S_i^{ESS} is the rated capacity of ESS at node i; $P_{t,i}^{ESS}$ and $Q_{t,i}^{ESS}$ denote the active power and reactive power output of ESS at node i at time interval t, respectively; $P_{t,i}^{ESS} > 0$ denotes the charging state and $P_{t,i}^{ESS} < 0$ denotes the discharging state; $P_i^{ESS,max}$ and $Q_i^{ESS,max}$ denote the upper limit of active power and reactive power of ESS, respectively; $SOC_{t,i}^{ESS}$ denotes the charging state of ESS connected to node i at time interval t; $SOC_i^{ESS,max}$ and $SOC_i^{ESS,min}$ denote the upper and lower limits of the charge state of ESS connected to node i.

As shown in the following figure, the flow chart of the proposed multi-timescale dynamic optimal dispatching method is as follows:



Flow chart of multi-time scale rolling optimization

The above multi-timescale scheduling model is second-order cone programming(SOCP) model.[61] It can be solved efficiently on a Raspberry Pi by invoking the Gurobi optimization toolbox via Python.

ADDITIONAL RESOURCES

Nomenclature

Parameters

r	leaking parameter	L	reservoir pool size
C	regularization parameter	I	unit matrix
M	input vector dimension	N	output vector dimension
N_0	number of initial base predictors	Γ	number of training set samples
n_0	number of top samples discarded	w_i^0	initial weight of the i-th sample
M_d	number of source domain set samples	N_d	number of target domain set samples
β	source domain weight parameter	D_L	setting low-dimensional dimension
$y_d^n(m)$	actual load value of the m th time interval on day n	N_1	number of leaking parameter candidate set
N_2	number of reservoir pool size candidate set	N_3	number of regularization parameter candidate set
k	sample size of correction factor	T_L	long-term forecasting days threshold
N_T	number of time intervals	N_B	number of nodes
R_{ij}	resistance of branch ij	X_{ij}	reactance of branch ij
V_{thr}^{max}, V_{thr}^{min}	expected upper and lower voltage limits	$\underline{V}, \overline{V}$	minimum and maximum voltage for safe operation

(*Continued on next page*)

Sets and vectors

$\boldsymbol{\Omega}$	training set	$\boldsymbol{\Omega}_s$	source domain data set
$\boldsymbol{\Omega}_t$	target domain data set	$\boldsymbol{\Omega}'$	normalized training set
L_a	the set of branches on the edge side	$\boldsymbol{\Omega}_a$	the set of nodes on the edge side
$\boldsymbol{x}(n)$	echo state vector for the time step n	$\boldsymbol{u}(n)$	input vector for the time step n
W_{in}	weight matrix connecting the input layer to the reservoir pool	W_{res}	weight matrix connecting the neurons in the reservoir pool
$\boldsymbol{y}(n)$	forecasting vector for the time step n	S	reservoir pool state matrix
Y	target output matrix	W_{out}	output matrix to be solved
D^0	initial sample weight distribution	$G_l(\boldsymbol{x})$	the l th base predictor
$y_d(i)$	actual value of the i th sample	$G(\boldsymbol{x})$	initial integrated forecasting model
$\boldsymbol{h}(n)$	low-dimensional state vector	H	low-dimensional state matrix
A	random weight matrix	B	random bias matrix
W_{enc}	encoding weight matrix	$G_{n,i}(\boldsymbol{x})$	the i-th base predictor on day n
$W_{out}^{n,i}$	output weight matrix of $G_{n,i}(\boldsymbol{x})$	$W_{out}^{n,i}(m)$	the m th row of $W_{out}^{n,i}$
$\boldsymbol{\epsilon}(n-1)$	systematic noise of state variables	$Q_{n-1}(m)$	covariance of $\boldsymbol{\epsilon}(n-1)$
$P_{n-1,i}(m)$	posterior estimation error covariance of $W_{out}^{n-1,i}(m)$	$Q_{n-1,i}(m)$	system noise covariance
$P_{n,i}^-(m)$	prior estimation error covariance of $W_{out}^{n,i}(m)$	$K_{n,i}(m)$	the i-th base predictor Kalman gain

Variables and functions

l	current iteration number	n	current data number
tanh()	hyperbolic tangent function	$\circ$	hadamard product
w_i^l	the i-th sample weight for the l-th iteration	D^l	sample weight distribution for l-th iteration

(*Continued on next page*)

Continued			
E_l	maximum regression error for the l-th iteration	$e_{l,i}$	the i-th relative regression error for the l-th iteration
e_l	overall regression error of $G_l(x)$	α_l	model weight coefficient of $G_l(x)$
β_l	error indicators calculated by e_l	R	model weight coefficient
p^*	optimal model weight moving step	h_1, h_2	key base predictor indexes
$\sigma(m)$	measurement noise of the m th time interval	$F_e(m)$	correction factor for the m th time interval
$V_{t,i}$	voltage amplitude at node i at time interval t	$I_{t,ij}$	current amplitude flowing between nodes i and j at time interval t
A_i	voltage deviation of node i	$P_{t,ji}, Q_{t,ji}$	active and reactive power flowing from node j to node i at time interval t
$P_{t,ik}, Q_{t,ik}$	active and reactive power flowing from node i to node k at time interval t	$P_{t,i}, Q_{t,i}$	net active power and net reactive power at node i at time interval t
$P_{t,i}^{DG}, Q_{t,i}^{DG}$	active and reactive power of DG at node i at time interval t	$P_{t,i}^{L}$, $Q_{t,i}^{L}$	active and reactive power of load at node i at time interval t
$P_{t,i}^{ESS}, Q_{t,i}^{ESS}$	active and reactive power output of ESS at node i at time interval t	$SOC_{t,i}^{ESS}$	charging state of ESS connected to node i at time interval t

Acronyms			
DGs	distributed generators	ADN	active distribution network
MG	microgrid	VPP	virtual power plant
SOP	soft open point	DSP	Digital Signal Processor
RF	Random Forest	GBRT	Gradient Boosting Regression Tree
AdaBoost	Adaptive Boosting	XGBoost	Extreme Gradient Boosting
MOPSO	multi-objective particle swarm optimization	CNNs	Convolution Neural Networks
SVR	Support Vector Regression	GB	Gradient Boosting
DT	Decision tree	BDI	Baltic Dry Index
ANNs	Artificial Neural Networks	ARIMA	Autoregressive integrated moving average
ELM	Extreme Learning Machine	KNN	K-Nearest Neighbor
PLS	Partial Least Squares Regression	RIDGE	Ridge regression
LSTM	Long Short-Term Memory	VMD	Variational Mode Decomposition
LSSVM	least squares support vector machines	ESN	echo state network
SSA	Singular Spectral Analysis	MPC	model predictive control
PCA	Principal component analysis	AE	Auto-encoder

Continued			
$\bar{I}$	maximum branch circuit current for safe operation	S_i^{DG}	DG capacity at node i
S_i^{ESS}	rated capacity of ESS at node i	$P_i^{ESS,max}$, $Q_i^{ESS,max}$	upper limit of active power and reactive power of ESS
$SOC_i^{ESS,max}$, $SOC_i^{ESS,min}$	upper and lower limits of the charge state of ESS connected to node i	λ_1, λ_2	weight coefficients of optimized subfunctions