https://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Articles in Advance, pp. 1–22
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

An Intelligent End-to-End Neural Architecture Search Framework for Electricity Forecasting Model Development

Jin Yang,[a] Guangxin Jiang,[a] Yinan Wang,[b] Ying Chen[a,*]

[a] School of Management, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China; [b] Department of Industrial and Systems Engineering, Rensselaer Polytechnic Institute, Troy, New York 12180

*Corresponding author

Contact: 22B910011@stu.hit.edu.cn, https://orcid.org/0000-0003-2741-138X (JY); gxjiang@hit.edu.cn, https://orcid.org/0000-0002-2604-7750 (GJ); wangy88@rpi.edu, https://orcid.org/0000-0002-4079-1658 (YW); yingchen@hit.edu.cn, https://orcid.org/0000-0002-0366-131X (YC)

Received: February 12, 2023
Revised: September 26, 2023; March 12, 2024; April 20, 2024
Accepted: April 22, 2024
Published Online in Articles in Advance: May 30, 2024

https://doi.org/10.1287/ijoc.2023.0034

Abstract. Recent years have witnessed exponential growth in developing deep learning models for time series electricity forecasting in power systems. However, most of the proposed models are designed based on the designers' inherent knowledge and experience without elaborating on the suitability of the proposed neural architectures. Moreover, these models cannot be self-adjusted to dynamically changed data patterns due to the inflexible design of their structures. Although several recent studies have considered the application of the neural architecture search (NAS) technique for obtaining a network with an optimized structure in the electricity forecasting sector, their training process is computationally expensive and their search strategies are not flexible, indicating that the NAS application in this area is still at an infancy stage. In this study, we propose an intelligent automated architecture search (IAAS) framework for the development of time series electricity forecasting models. The proposed framework contains three primary components, that is, network function–preserving transformation operation, reinforcement learning–based network transformation control, and heuristic network screening, which aim to improve the search quality of a network structure. After conducting comprehensive experiments on two publicly available electricity load data sets and two wind power data sets, we demonstrate that the proposed IAAS framework significantly outperforms the 10 existing models or methods in terms of forecasting accuracy and stability. Finally, we perform an ablation experiment to showcase the importance of critical components in the proposed IAAS framework in improving forecasting accuracy.

History: Accepted by Ram Ramesh, Area Editor for Data Science and Machine Learning.
Funding: J. Yang, G. Jiang, and Y. Chen were supported by the National Natural Science Foundation of China [Grants 72293562, 72121001, 72101066, 72131005, 71801148, and 72171060]. Y. Chen was supported by the Heilongjiang Natural Science Excellent Youth Fund [YQ2022G004].
Supplemental Material: The software (Yang et al. 2023) that supports the findings of this study is available within the paper and its Supplemental Information (https://pubsonline.informs.org/doi/suppl/10.1287/ijoc.2023.0034) as well as from the IJOC GitHub software repository (https://github.com/INFORMSJoC/2023.0034). The complete IJOC Software and Data Repository is available at https://informsjoc.github.io/.

Keywords: neural architecture search • electricity forecasting • recurrent neural network • reinforcement learning • network transformation

1. Introduction

1.1. Overview

The electricity market has reshaped the electricity trade mode since the introduction of competitive market and deregulation processes during the early 1990s (Weron 2014). As electricity is a tradeable commodity that cannot be stored on a large scale, modern electricity markets require a balance between electricity production and consumption in real time (Gan et al. 2020). Consequently, this requirement plays an integral role in maintaining the stable operations and regulations of power systems (Huang et al. 2021). However, there are several factors that affect this balance.

From the production perspective, many renewable energy sources, such as wind and solar, are increasingly contributing to the power systems with rapid growth (Solaun and Cerdá 2019). According to the fuel report from 2021 International Energy Agency, the overall global renewable electricity is predicted to be more than 4,800 gigawatts (GW) in 2026, which is equivalent to the total power capacity of fossil fuels and nuclear combined in 2020.

Compared with traditional energy resources, for example, coal and gas, renewable energy is sustainable and clean (Munawer 2018, Nyashina et al. 2020). However, its uncertain and intermittent nature brings significant challenges to the smooth and secure operation of power systems (Pryor et al. 2020, Chen et al. 2023). From the consumption perspective, electricity load can be correlated with various patterns related to industrial activities and weather conditions (Chen et al. 2018, Jalali et al. 2021a). For instance, industrial and commercial consumers usually consume more electricity during daytime than at night and use more electricity during summer than in spring or autumn. Nonetheless, these generic electricity use patterns on any given day are still full of uncertainties (Billings et al. 2022, Li et al. 2022) because most consumption behaviors are uncontrollable. To enhance the secure and reliable operation, electricity forecasting has become one of the most effective techniques to minimize these uncertainties in modern power systems (Sunar and Birge 2019).

Contrasting with numerous other commodities, electricity entails being consumed immediately after being generated (Huang et al. 2021), which brings high requisites in forecasting accuracy. As artificial intelligence (AI) techniques are demonstrating outstanding performance in predicting and capturing uncertainties, researchers have turned to using advanced machine learning (ML) methods instead of traditional statistical time series methods (e.g., autoregressive moving average and exponential smoothing) for the electricity forecasting model development. For example, Hu et al. (2015) used support vector machine (SVM) to forecast wind power generation; Liu and Sun (2019) used random forest (RF) method to predict solar power generation; Islam et al. (2014) exploited artificial neural network (ANN) to forecast load demand. Various other relevant research studies can be found in the literature, such as Chen et al. (2013), Lahouar and Slama (2015), Shepero et al. (2018), Srivastava et al. (2019), and so on. On the other hand, the remarkable success of deep learning (DL) can be witnessed in pattern recognition, object detection, sales forecasting, and other applications (Zhou et al. 2021b, Bi et al. 2022, Hu and Hong 2022, Liu et al. 2022, Zhang et al. 2022). Therefore, we have seen a dramatic trend in using DL methods in recent years to develop forecasting models for power systems. For instance, in wind power forecasting, Shahid et al. (2021) introduced a novel genetic long short-term memory (LSTM) network model, and Xiong et al. (2022) proposed DL models based on attention mechanism. In solar power forecasting, Heo et al. (2021) introduced a multichannel convolutional neural network (CNN) model, and Agga et al. (2021) proposed two models: one is CNN+LSTM model, and the other is the convolutional LSTM model. In load forecasting, Chen et al. (2018) introduced two deep residual network models, and Jalali et al. (2021a) proposed an evolutionary-based deep CNN model. Compared with traditional ML methods (e.g., SVM and RF), DL models have more flexible structures. Consequently, these models are likely to capture the hidden patterns within data and then construct powerful forecasting models.

The primary goal of this research study is to provide a robust and end-to-end framework that can flexibly self-adjust the deep neural network structures for adapting to various data sets to produce high-quality forecasting models for power systems. There are two motivations behind our considerations. First, numerous prior research studies have designed model structures based on their knowledge or experience (Elsken et al. 2019), especially in the domain of energy forecasting. For example, Shahid et al. (2021) designed a three-layer LSTM model for wind power forecasting; Heo et al. (2021) developed a four-layer multichannel CNN model for solar power forecasting; Jalali et al. (2021a) presented a model containing two CNN layers, a pooling layer and a fully connected network (FCN) layer, for load forecasting. Although these research studies have conducted abundant experiments to demonstrate the advantages of their proposed models, they fail to clarify the rationales for designing each specific model structure. As noted, a larger (or deeper) network would usually achieve a better performance than a smaller network (Ng et al. 2015) without considering the overfitting issue. In other words, whether a particular forecasting model could be further improved by using more layers (e.g., CNN or LSTM) or including more units in each layer is still unknown. As described in Ren et al. (2021), designing an optimal neural architecture only based on the inherent knowledge of human beings is problematic, because it is difficult for people to jump out of their thinking paradigms. The second motivation is that, although there are some initial trials on designing network structures, all these methods either only optimize the network structure incrementally (incapable of pruning) or stack the newly added layers in a certain paradigm. For example, in Jalali et al. (2021b), a heuristic algorithm is used to optimize the number of CNN and LSTM layers following the fixed structure that CNN layers are built on the top of LSTM layers; in Shahid et al. (2021), a genetic algorithm is used to only optimize the number of neurons in LSTM networks without considering other network structures (e.g., CNN and RNN) in the forecasting model. Such a paradigm simplifies the optimization problem while inevitably decreasing the flexibility of the network structure search. Many factors, such as climate conditions and industrial activities, will introduce significant uncertainties and intermittency to power systems, resulting in a dynamic pattern. Therefore, an automated neural architecture optimization algorithm is required to be more flexible for power systems such that it can adaptively generate high-quality and self-adjusted forecasting models.

1.2. Literature Review

Neural architecture search (NAS) is a technique that can help automate the architecture design of neural networks (Elsken et al. 2019). In the computer vision area, numerous NAS techniques have been proposed, which include intermittent-aware NAS (Mendis et al. 2021), instance-aware NAS (Cheng et al. 2020), and platform-aware NAS (Tan et al. 2019). These techniques have exhibited promising and competitive results on some public benchmark data sets. However, in the domain of time series forecasting, only a few works have been reported. Limited examples include a study by Pan et al. (2021), which designed a NAS method for predictions of air quality and traffic speed. Based on Pan et al. (2021), Chen et al. (2021) developed a scale-aware NAS (SNAS) method for multivariate time series forecasting, where the gradient-based optimization method was used in the search process. Regarding electricity forecasting, Khodayar et al. (2017) developed an NAS strategy based on the rough set theory (Pawlak 1982) for short-term wind speed forecasting, and Torres et al. (2019) used a random model based on the NAS strategy for load forecasting. Nonetheless, both NAS strategies were regarded as unintelligent and inefficient in Jalali et al. (2021b), which then proposed an improved evolutionary whale optimization algorithm to optimize the neural architecture for wind power forecasting. Although the latest NAS techniques in Chen et al. (2021) and Jalali et al. (2021b) have achieved some success in the time series model development, two issues in these studies have been ignored, namely high computational cost and limited search flexibility. In the following, we conduct a literature review in terms of these two issues in NAS, and then present the knowledge gaps.

1.2.1. High Computational Cost. In terms of high computational cost, reinforcement learning (RL)-based search methods are used to provide a more controllable search space (Baker et al. 2017, Zoph and Le 2017) than evolutionary-based methods (Real et al. 2017, Xie and Yuille 2017, Jalali et al. 2021b), which means RL-based methods can reduce the computational cost. Nonetheless, such methods are also limited by the computational issues because each sampled architecture has to be trained from the beginning (Baymurzina et al. 2022). Concerning this, many research works have proposed warm-started methods by inheriting the weights of the existing network. A representative example is the efficient architecture search (EAS) framework (Cai et al. 2018). Briefly, Cai et al. (2018) used the Net2Net function-preserving transformation framework (Chen et al. 2016) and proposed an RL-based EAS framework to widen or deepen an existing network efficiently without modifying the mapping implemented by the network. As noted, the Net2Net framework (Chen et al. 2016) is likely to rapidly transfer knowledge stored in one neural network to another and contains widening and deepening operations to conduct network function-preserving transformations for CNN and FCN. Despite the excellent performance of the EAS framework, directly applying it to time series data are inapplicable for two reasons: (i) the CNN and FCN structures cannot well address the internal temporal dependency within time series data; and (ii) the function-preserving transformations for recurrent neural network (RNN) or LSTM (a special case of RNN) have not been investigated.

1.2.2. Limited Search Flexibility. In terms of limited search flexibility, existing NAS studies cannot search for neural architectures from two directions. Taking an example of the EAS framework (Cai et al. 2018), the updated models can only be enlarged (widened or deepened) so that the models obtained are vulnerable to overfitting the training data (Liu et al. 2018). Contrary to the EAS framework, the N2N framework (Ashok et al. 2017) proposed a reverse operation that compresses networks using RL by layer removal and layer shrinkage. However, this framework is designed to compress or prune existing networks such as ResNet-18 and VGG-19, rather than directly building the model from data. Because of both limitations of the existing works, how to develop a search strategy that cannot only enlarge the networks but also shrink them becomes an effective way to increase the search flexibility in NAS.

To fill in the research gaps, in this study, we adapt the EAS framework to time series data to address the computational issue in NAS. Moreover, we develop a flexible search strategy that modifies the network structures from two directions to identify a high-quality electricity time series forecasting model. The contributions of our proposed framework are discussed in the following section.

1.3. Contributions

In this study, we propose an intelligent automated architecture search (IAAS) framework, which contains three main components, that is, network transformation operation, network transformation control, and network heuristic screening. The first two components jointly generate new architectures of the given model. The third component expands the search space by using multiple network structures as candidates and screens the generated architectures to only preserve the ones that lead to better performance. With such a framework, we can build a high-quality forecasting model for the given electricity time series data. The key technical contributions of the proposed IAAS framework have three aspects.

Our first contribution is to adapt the EAS framework to time series data. As discussed previously, the EAS framework (Cai et al. 2018) has solved the computational problem in NAS by adopting the Net2Net framework (Chen et al. 2016). However, the EAS framework targets image data, which is different from time series electricity data. Therefore, to better analyze the electricity data and extract features from them, we consider RNN as well as CNN and FCN in network transformation operation. However, function-preserving transformation methods for RNN are lacking in the literature. Directly applying the transformation formulations (Chen et al. 2016) for spatial networks (e.g., CNN and FCN) to RNN is not appropriate because the current formulations for these networks do not consider the inherent temporal information. Therefore, we innovatively develop the RNN function-preserving transformation via the following steps: (i) we propose a general feature learning formulation for all types of neural networks; (ii) the RNN feature learning process is redesigned based on the proposed general formulation; and (iii) we propose wider and deeper function-preserving transformation methods for the reformulated RNN structures. Additionally, considering the outstanding performance of LSTM in developing the time series forecasting models (Agga et al. 2021, Shahid et al. 2021), we also introduce the function-preserving transformation for LSTM following a similar logic.

Our second contribution is to improve the search flexibility of network structures. The EAS framework (Cai et al. 2018) exploits the RL-based meta-controllers to modify the existing network. In the transformation control, the RL-based meta-controllers in EAS are predetermined to take five steps of deeper transformation and four steps of wider transformation at each search episode. This hard-coded transformation order cannot be intelligently changed, and the search process is not flexible. As a comparison, the network operation control in our IAAS framework has two advantages: (i) besides the wider actor and deeper actor used to enlarge the networks, the IAAS framework has a pruning process to shrink the network structures; and (ii) we introduce a selector actor to automatically determine whether to widen, deepen, prune or keep the network unchanged at each search episode instead of using the fixed order of the network transformation operations as conducted in EAS. The first advantage is that the search is now bidirectional (increase/decrease), improving the search flexibility. The second advantage is that it avoids the hard-coded transformation, which enables intelligent and adaptive network structure transformation. In addition, we also modify the meta-controllers (i.e., wider actor and deeper actor) from the EAS framework so that the IAAS framework can widen and deepen the network more intelligently and flexibly without following a fixed paradigm that the CNN layers have to be built on the top of FCN layers (Cai et al. 2018).

Our third contribution is to expand the search space of network structures to improve the possibility of obtaining high-quality solutions. In EAS, the new network architecture generated by the transformations is directly used in the next-round transformation without screening. Such an implementation is like using one trajectory to train the RL-based meta-controllers. This potentially has one problem: One fixed-length trajectory may not collect enough samples to train the RL-based meta-controllers within limited computational resources. Therefore, some wrong actions of RL may occur during the search procedure. To remedy this, we propose a net pool module in IAAS. Benefiting from this module, we can expand the search space using multiple network structure candidates in the net pool. Moreover, we propose a heuristic screening algorithm to manage the network structures in the net pool so that IAAS can iteratively transform a set of networks for continuous performance improvement.

We perform comprehensive experiments to evaluate our approach based on two different cases. One is for wind power forecasting from the electricity production side, and the other is for load forecasting from the electricity consumption side. The wind power data sets have been acquired from a wind power company in China, and the load data sets have been collected from ISO New England Inc.[1] The experiments mainly contain two parts: comparison experiments and ablation experiments. In the comparison experiment, we use 10 existing models or methods with standard performance metrics to demonstrate the effectiveness of our framework. As to the ablation experiments, the selector actor and net pool are new compared with the EAS framework. Therefore, we develop three variants of the IAAS framework to showcase the significance of proposing selector actor and net pool components in the IAAS framework.

All in all, we make the following contributions to the literature:

(i) We propose an RL-based IAAS framework, aiming to build a high-quality electricity forecasting model;

(ii) We reformulate the feature learning operation of RNN and LSTM structures, and propose the function-preserving transformation methods for each of them;

(iii) We propose a network transformation control system with RL so that the network structures can both be intelligently and flexibly enlarged and shrunk;

(iv) We propose a heuristic screening algorithm for the net pool component to iteratively improve the quality of the searched neural architectures; and

(v) We conduct extensive comparison experiments and ablation experiments on existing electricity data sets to explore the performance of the proposed IAAS framework.

1.4. Organization of This Article

The remainder of this paper has been organized as follows. Section 2 provides the network transformation methods, including function-preserving transformations for RNN and LSTM and a shrinkage transformation. Section 3 proposes the IAAS framework and details its key components. Section 4 demonstrates the advantages of the proposed IAAS framework via comparison and ablation experiments. Finally, in Section 5, we discuss the extension of the IAAS framework and provide the concluding remarks.

2. Network Transformation

In the process of searching for a high-quality network structure, we consider two methods to transform the network: one is a function-preserving transformation method to enlarge the network and the other is a pruning method to shrink the network.

Function-preserving transformation indicates that the change in the network architecture should not influence its functionality (Chen et al. 2016). Thus, the initial values for the new set of network parameters $\boldsymbol{\theta}'$ in the student neural network $G(\boldsymbol{x};\boldsymbol{\theta}')$ (i.e., the extended neural network) should be determined to ensure it has the same outputs as the teacher neural network $F(\boldsymbol{x};\boldsymbol{\theta})$ (i.e., the original neural network) with the same input $\boldsymbol{x}$. The function-preserving transformation is formulated by the following equation:

$$\forall \boldsymbol{x}, F(\boldsymbol{x};\boldsymbol{\theta}) = G(\boldsymbol{x};\boldsymbol{\theta}'). \tag{1}$$

Equation (1) is an essential characteristic in our proposed framework, indicating that after the network's wider or deeper transformations, the outputs of the student network are the same as those of the teacher network. This property ensures the student network inherits the experience learned by the teacher network and thus enables the warm start of its training process to improve the training efficiency. We first reformulate RNN and LSTM into a two-phase feature learning process in Online Appendix A. Subsequently, we investigate the function-preserving transformation to widen and deepen the RNN and LSTM in Sections 2.1 and 2.2, respectively. After that, we present the detailed information of how to prune the network in Section 2.3.

2.1. Wider Transformation

The wider transformation of a neural network influences at least two layers, which makes it complicated. We will first introduce the wider transformation in RNN and then extend it to LSTM. Suppose that an RNN layer i has p_i input and p_{i+1} output features, and RNN layer $i+1$ has p_{i+2} output features, then the parameters of layers i and $i+1$ are $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i \times p_{i+1}}$, $\boldsymbol{H}^{(i)} \in \mathbb{R}^{p_{i+1} \times p_{i+1}}$ and $\boldsymbol{W}^{(i+1)} \in \mathbb{R}^{p_{i+1} \times p_{i+2}}$, $\boldsymbol{H}^{(i+1)} \in \mathbb{R}^{p_{i+2} \times p_{i+2}}$, respectively. If we widen layer i to layer i' with p'_{i+1} output features ($p'_{i+1} > p_{i+1}$), then the original network parameters of layers i and $i+1$ are replaced by $\boldsymbol{W}^{(i')} \in \mathbb{R}^{p_i \times p'_{i+1}}$, $\boldsymbol{H}^{(i')} \in \mathbb{R}^{p'_{i+1} \times p'_{i+1}}$ and $\boldsymbol{W}^{(i'+1)} \in \mathbb{R}^{p'_{i+1} \times p_{i+2}}$, $\boldsymbol{H}^{(i'+1)} \in \mathbb{R}^{p_{i+2} \times p_{i+2}}$, separately.

To derive the wider transformation for RNN, we follow Chen et al. (2016) to specify a random mapping function $g: \{1,2,\ldots,p'_{i+1}\} \to \{1,2,\ldots,p_{i+1}\}$ that satisfies:

$$g(k) = \begin{cases} k & k \le p_{i+1}, \\ \text{random sample from } 1,2,\ldots,p_{i+1} & k > p_{i+1}. \end{cases} \tag{2}$$

Equation (2) indicates how the features in a new student network layer have been represented. The first p_{i+1} features are exactly the same as those in its teacher network layer, whereas from the $(p_{i+1}+1)th$ feature, they have been randomly replicated from the original features in the teacher network layer. To keep the function-preserving transformation, we use a replication factor, f_w, similar to that in Chen et al. (2016):

$$f_w(k) = \frac{1}{|\{l|g(k)=g(l)\}|}, l = 1,2,\ldots,p_{i+1}, k = 1,2,\ldots,p'_{i+1}, \tag{3}$$

where $|\{l|g(k)=g(l)\}|$ is the cardinality of set $\{l|g(k)=g(l)\}$. This term represents the replication number of one output feature in the original network. For example, if a feature is replicated once, there will be two exactly identical features in the output vector. Then, the replication factor is $1/2$, which will be employed simultaneously as the replicated feature weights and its original feature. Subsequently, we can derive the RNN wider transformation with the new student network layer parameters specified by Proposition 1 and its corresponding proof can be found in Online Appendix B.

Proposition 1. *When widening a layer i to a layer i' in RNN, we can set the parameters of layer $i'$and layer $i'+1$, $\boldsymbol{W}^{(i')}$, $\boldsymbol{H}^{(i')}$, $\boldsymbol{W}^{(i'+1)}$, and $\boldsymbol{H}^{(i'+1)}$, as*

$$\begin{aligned}
W^{(i')}_{j,k} &= W^{(i)}_{j,g(k)}, && j=1,2,\dots,p_i, k=1,\dots,p'_{i+1}\\
H^{(i')}_{l,k} &= f_w(l)H^{(i)}_{g(l),g(k)}, && l=1,2,\dots,p'_{i+1}, k=1,\dots,p'_{i+1}\\
W^{(i'+1)}_{k,h} &= f_w(k)W^{(i+1)}_{g(k),h}, && k=1,2,\dots,p'_{i+1}, h=1,\dots,p_{i+2}\\
H^{(i'+1)}_{r,h} &= H^{(i+1)}_{r,h}, && r=1,2,\dots,p_{i+2}, h=1,\dots,p_{i+2}.
\end{aligned}$$

Then, the wider function-preserving transformation for RNN layer i is satisfied.

For LSTM, we also assume that LSTM layer i has p_i input and p_{i+1} output features, and LSTM layer $i+1$ has p_{i+1} input and p_{i+2} output features. Hence, the parameters of layer i are $\boldsymbol{W}^{(i)}_f \in \mathbb{R}^{p_i\times p_{i+1}}$, $\boldsymbol{W}^{(i)}_i \in \mathbb{R}^{p_i\times p_{i+1}}$, $\boldsymbol{W}^{(i)}_o \in \mathbb{R}^{p_i\times p_{i+1}}$, $\boldsymbol{W}^{(i)}_c \in \mathbb{R}^{p_i\times p_{i+1}}$, $\boldsymbol{H}^{(i)}_f \in \mathbb{R}^{p_{i+1}\times p_{i+1}}$, $\boldsymbol{H}^{(i)}_i \in \mathbb{R}^{p_{i+1}\times p_{i+1}}$, $\boldsymbol{H}^{(i)}_o \in \mathbb{R}^{p_{i+1}\times p_{i+1}}$, and $\boldsymbol{H}^{(i)}_c \in \mathbb{R}^{p_{i+1}\times p_{i+1}}$; moreover, the parameters of layer $i+1$ are $\boldsymbol{W}^{(i+1)}_f \in \mathbb{R}^{p_{i+1}\times p_{i+2}}$, $\boldsymbol{W}^{(i+1)}_i \in \mathbb{R}^{p_{i+1}\times p_{i+2}}$, $\boldsymbol{W}^{(i+1)}_o \in \mathbb{R}^{p_{i+1}\times p_{i+2}}$, $\boldsymbol{W}^{(i+1)}_c \in \mathbb{R}^{p_{i+1}\times p_{i+2}}$, $\boldsymbol{H}^{(i+1)}_f \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$, $\boldsymbol{H}^{(i+1)}_i \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$, $\boldsymbol{H}^{(i+1)}_o \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$, and $\boldsymbol{H}^{(i+1)}_c \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$. If we widen layer i to layer i' with p'_{i+1} output features ($p'_{i+1} > p_{i+1}$), the original network parameters of layers i are replaced by $\boldsymbol{W}^{(i')}_f \in \mathbb{R}^{p_i\times p'_{i+1}}$, $\boldsymbol{W}^{(i')}_i \in \mathbb{R}^{p_i\times p'_{i+1}}$, $\boldsymbol{W}^{(i')}_o \in \mathbb{R}^{p_i\times p'_{i+1}}$, $\boldsymbol{W}^{(i')}_c \in \mathbb{R}^{p_i\times p'_{i+1}}$, $\boldsymbol{H}^{(i')}_f \in \mathbb{R}^{p'_{i+1}\times p'_{i+1}}$, $\boldsymbol{H}^{(i')}_i \in \mathbb{R}^{p'_{i+1}\times p'_{i+1}}$, $\boldsymbol{H}^{(i')}_o \in \mathbb{R}^{p'_{i+1}\times p'_{i+1}}$, and $\boldsymbol{H}^{(i')}_c \in \mathbb{R}^{p'_{i+1}\times p'_{i+1}}$. In addition, the parameters of layer $i+1$ are replaced by $\boldsymbol{W}^{(i'+1)}_f \in \mathbb{R}^{p'_{i+1}\times p_{i+2}}$, $\boldsymbol{W}^{(i'+1)}_i \in \mathbb{R}^{p'_{i+1}\times p_{i+2}}$, $\boldsymbol{W}^{(i'+1)}_o \in \mathbb{R}^{p'_{i+1}\times p_{i+2}}$, $\boldsymbol{W}^{(i'+1)}_c \in \mathbb{R}^{p'_{i+1}\times p_{i+2}}$, $\boldsymbol{H}^{(i'+1)}_f \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$, $\boldsymbol{H}^{(i'+1)}_i \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$, $\boldsymbol{H}^{(i'+1)}_o \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$, and $\boldsymbol{H}^{(i'+1)}_c \in \mathbb{R}^{p_{i+2}\times p_{i+2}}$. Therefore, we can derive the LSTM wider transformation with the new student network layer parameters specified by Proposition 2, and its proof can be found in Online Appendix B.

Proposition 2. *When widening a layer i to a layer i' in LSTM, we can set the parameters of layer $i'$and layer $i'+1$ as*

$$\begin{aligned}
W^{(i')}_{f;j,k} &= W^{(i)}_{f;j,g(k)}, && j=1,2,\dots,p_i, k=1,\dots,p'_{i+1}\\
W^{(i')}_{i;j,k} &= W^{(i)}_{i;j,g(k)}, && j=1,2,\dots,p_i, k=1,\dots,p'_{i+1}\\
W^{(i')}_{o;j,k} &= W^{(i)}_{o;j,g(k)}, && j=1,2,\dots,p_i, k=1,\dots,p'_{i+1}\\
W^{(i')}_{c;j,k} &= W^{(i)}_{c;j,g(k)}, && j=1,2,\dots,p_i, k=1,\dots,p'_{i+1}\\
H^{(i')}_{f;l,k} &= f_w(l)H^{(i)}_{f;g(l),g(k)}, && l=1,2,\dots,p'_{i+1}, k=1,\dots,p'_{i+1}\\
H^{(i')}_{i;l,k} &= f_w(l)H^{(i)}_{i;g(l),g(k)}, && l=1,2,\dots,p'_{i+1}, k=1,\dots,p'_{i+1}\\
H^{(i')}_{o;l,k} &= f_w(l)H^{(i)}_{o;g(l),g(k)}, && l=1,2,\dots,p'_{i+1}, k=1,\dots,p'_{i+1}\\
H^{(i')}_{c;l,k} &= f_w(l)H^{(i)}_{c;g(l),g(k)}, && l=1,2,\dots,p'_{i+1}, k=1,\dots,p'_{i+1}\\
W^{(i'+1)}_{f;k,h} &= f_w(k)W^{(i+1)}_{f;g(k),h}, && k=1,2,\dots,p'_{i+1}, h=1,\dots,p_{i+2}\\
W^{(i'+1)}_{i;k,h} &= f_w(k)W^{(i+1)}_{i;g(k),h}, && k=1,2,\dots,p'_{i+1}, h=1,\dots,p_{i+2}\\
W^{(i'+1)}_{o;k,h} &= f_w(k)W^{(i+1)}_{o;g(k),h}, && k=1,2,\dots,p'_{i+1}, h=1,\dots,p_{i+2}\\
W^{(i'+1)}_{c;k,h} &= f_w(k)W^{(i+1)}_{c;g(k),h}, && k=1,2,\dots,p'_{i+1}, h=1,\dots,p_{i+2}\\
H^{(i'+1)}_{f;r,h} &= H^{(i+1)}_{f;r,h}, && r=1,2,\dots,p_{i+2}, h=1,\dots,p_{i+2}\\
H^{(i'+1)}_{i;r,h} &= H^{(i+1)}_{i;r,h}, && r=1,2,\dots,p_{i+2}, h=1,\dots,p_{i+2}\\
H^{(i'+1)}_{o;r,h} &= H^{(i+1)}_{o;r,h}, && r=1,2,\dots,p_{i+2}, h=1,\dots,p_{i+2}\\
H^{(i'+1)}_{c;r,h} &= H^{(i+1)}_{c;r,h}, && r=1,2,\dots,p_{i+2}, h=1,\dots,p_{i+2}.
\end{aligned}$$

Then, the wider function-preserving transformation for LSTM layer i is satisfied.

2.2. Deeper Transformation

The deeper transformations for RNN and LSTM are developed in a much more straightforward manner than their wider transformations. In the following, we will first use RNN to illustrate the essential process of deeper transformation and then extend it to LSTM. Similar to Chen et al. (2016), the deeper transformation should replace an RNN layer with two RNN layers using the identity mapping method. In other words, the input features of the new student RNN layer should be identical to those of the teacher RNN layer. Suppose that we deepen an RNN layer i with a new student layer i'' by inserting the student layer i'' after the RNN layer i. If layer i has p_{i+1} output features, then layer i'' is assumed to have p_{i+1} input and p_{i+1} output features for simplicity. The input and output sizes of layer i'' are not restricted to be p_{i+1}. Using the other value is also acceptable. However, this brings more complexities to guarantee the function-preserving transformation, which is beyond the scope of this study. The new student network parameters of the layer i'' are $\boldsymbol{W}^{(i'')} \in \mathbb{R}^{p_{i+1}\times p_{i+1}}$ and $\boldsymbol{H}^{(i'')} \in \mathbb{R}^{p_{i+1}\times p_{i+1}}$. According to Chen et al. (2016), the deeper transformation is applicable when the activation function has this property, that is, $f_a(f_a(z)) = f_a(z)$, where z is a vector. This property means that, although a vector passes the activation function multiple times, the output is kept unchanged. Considering this, we employ the rectified linear unit (ReLU) activation function to satisfy this property. In addition, for a more general application that the network has no RNN layers, we are also able to insert a new student RNN layer into the network if the new student network parameters satisfy Proposition 3. For example, in a network, if layer i is a CNN layer and layer $i+1$ is an FCN layer, we can insert an RNN layer between layer i and layer $i+1$.

Proposition 3. *When deepening layer i with an RNN layer i'' in a network, we can set the RNN layer parameters* $W^{(i'')}_{l,k}$ *and* $H^{(i'')}_{l,k}$ *as*

$$W^{(i'')}_{l,k} = \begin{cases} 1 & l = k \\ 0 & l \neq k \end{cases}, \qquad l = 1,2,\ldots,p_{i+1}, k = 1,\ldots,p_{i+1},$$

$$H^{(i'')}_{l,k} = 0, \qquad l = 1,2,\ldots,p_{i+1}, k = 1,\ldots,p_{i+1}.$$

Then, the deeper function-preserving transformation for RNN layer i is satisfied.

For LSTM, we can insert a new student LSTM layer into the network if the parameters of this new student network fulfill Proposition 4. Similar to RNN, LSTM can be inserted into any place of a network.

Proposition 4. *When deepening layer i with an LSTM layer i'' in a network, we can set the LSTM layer parameters as*

$$W^{(i'')}_{o;l,k} = \begin{cases} 1 & l = k \\ 0 & l \neq k \end{cases}, \qquad l = 1,2,\ldots,p_{i+1}, k = 1,\ldots,p_{i+1},$$

$$W^{(i'')}_{f;l,k} = W^{(i'')}_{i;l,k} = W^{(i'')}_{c;l,k} = 0, \qquad l = 1,2,\ldots,p_{i+1}, k = 1,\ldots,p_{i+1},$$

$$H^{(i'')}_{f;l,k} = H^{(i'')}_{i;l,k} = H^{(i'')}_{o;l,k} = H^{(i'')}_{c;l,k} = 0, \qquad l = 1,2,\ldots,p_{i+1}, k = 1,\ldots,p_{i+1}.$$

Then, the deeper function-preserving transformation for LSTM layer i is satisfied.

2.3. Shrinkage Transformation

In this study, we consider a score-based pruning method, that is, movement pruning (Sanh et al. 2020), to shrink the network structures. Score-based pruning is to prune the network based on the importance score. In the following, we use FCN as an example to illustrate how the movement pruning method performs. Let $\boldsymbol{W}^{(i)} \in \mathbb{R}^{p_i\times p_{i+1}}$ be the weight matrix for layer i and $\boldsymbol{x}^{(i+1)} = \boldsymbol{W}^{(i)}\boldsymbol{x}^{(i)} \in \mathbb{R}^{p_{i+1}}$ be the output for the input $\boldsymbol{x}^{(i)} \in \mathbb{R}^{p_i}$. To determine which weights to be pruned, we use the importance scores $\boldsymbol{\Lambda}^{(i)} \in \mathbb{R}^{p_i\times p_{i+1}}$ for $\boldsymbol{W}^{(i)}$.

According to $\boldsymbol{\Lambda}^{(i)}$, the pruning strategy is built on a mask $\boldsymbol{M}^{(i)} \in \{0,1\}^{p_i\times p_{i+1}}$, which generates a new weight matrix $\boldsymbol{W}^{(i)} \odot \boldsymbol{M}^{(i)}$ ($\odot$ is the Hadamard product), to produce a new output $\boldsymbol{x}^{(i+1)} = (\boldsymbol{W}^{(i)} \odot \boldsymbol{M}^{(i)})\boldsymbol{x}^{(i)}$. Particularly, Sanh et al. (2020) defines the Top_v strategy to compute the mask, which selects the v% highest values in $\boldsymbol{\Lambda}^{(i)}$, as presented in Equation (4):

$$Top_v(\boldsymbol{\Lambda}^{(i)})_{l,k} = \begin{cases} 1, & \Lambda^{(i)}_{l,k}\ in\ top\ v\%\ , l = 1,\ldots,p_i, k = 1,\ldots,p_{i+1}, \\ 0, & otherwise, \end{cases} \tag{4}$$

where $\Lambda^{(i)}_{l,k}, l = 1,\ldots,p_i, k = 1,\ldots,p_{i+1}$, represent one element in $\boldsymbol{\Lambda}^{(i)}$.

During training, the movement pruning method uses the first-order information of the loss function L to update the importance score $\Lambda_{l,k}^{(i)}$ with the gradient:

$$\frac{\partial L}{\partial \Lambda_{l,k}^{(i)}} = \frac{\partial L}{\partial x_k^{(i+1)}} \frac{\partial x_k^{(i+1)}}{\partial \Lambda_{l,k}^{(i)}} = \frac{\partial L}{\partial x_k^{(i+1)}} W_{l,k}^{(i)} x_l^{(i)}, l = 1, \dots, p_i, k = 1, \dots, p_{i+1}. \tag{5}$$

Equation (5) implies that the importance score $\Lambda_{l,k}^{(i)}$ increases when $W_{l,k}^{(i)}$ moves away from zero and decreases when $W_{l,k}^{(i)}$ moves toward to zero. The gradient of importance score is calculated by automatic differentiation technique (Baydin et al. 2018). In this study, we consider the weight associated with a particular neuron to be the smallest unit for executing the pruning operation. We calculate the importance score of the output neuron k as

$$\lambda_k^{(i+1)} = \sum_{l=1}^{p_i} \Lambda_{l,k}^{(i)}, \tag{6}$$

where $\lambda_k^{(i+1)}$ represent the kth element in vector $\boldsymbol{\lambda}^{(i+1)}$, the importance score for the output $\boldsymbol{x}^{(i+1)}$ is represented as $\boldsymbol{\lambda}^{(i+1)}$, and the mask for $\boldsymbol{\lambda}^{(i+1)}$ is represented as $\boldsymbol{m}^{(i+1)}$. We can then calculate the importance score of all neurons and use the Top_v strategy to eliminate neurons of less importance, as well as their corresponding weights. Consequently, we can obtain a smaller network structure. In Online Appendix C, we present the details of how to use the movement pruning method to prune FCN, RNN, CNN, and LSTM.

3. IAAS Framework Development

In this section, we propose an IAAS framework to intelligently and flexibly search the neural architecture. Primarily, this framework has three main components, that is, network transformation operation, network transformation control, and network heuristic screening. The new model architecture is generated by conducting the network transformation operation determined by the network transformation control. Subsequently, the newly generated model architectures are further evaluated and selected by the network heuristic screening. In the following, we first describe the overview of the IAAS framework and then introduce three actor networks and pruning implementation details as well as the net pool component used in the framework.

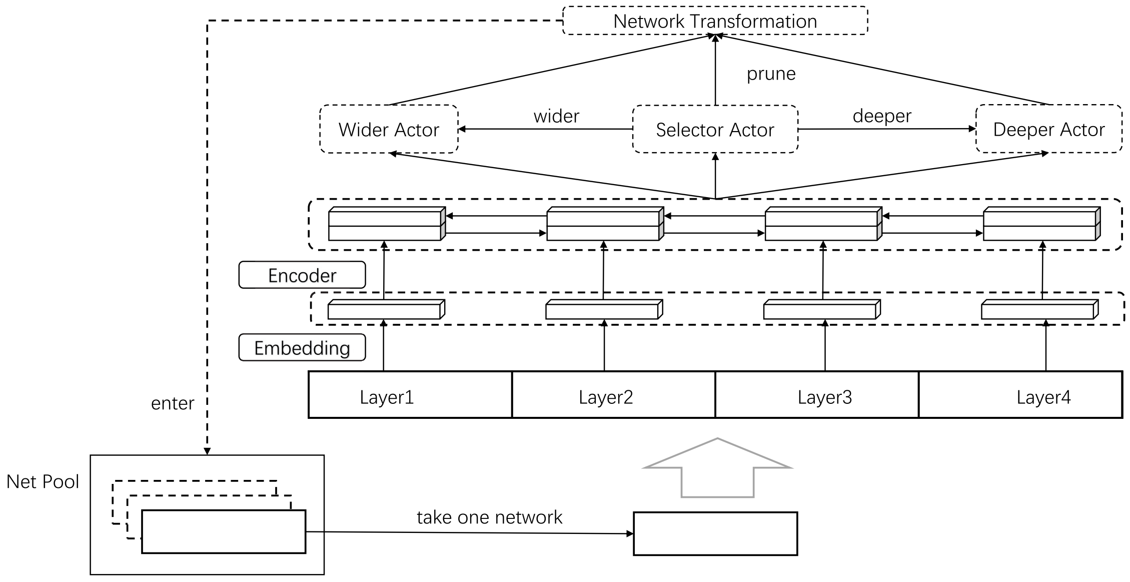
3.1. Overview of IAAS Framework

We design the IAAS framework with an end-to-end mode. Specifically, we do not need to manually determine the number of layers and the size of each layer. Moreover, this framework can automatically determine when to widen, deepen, and prune the network. Last, after the maximum number of search episodes is reached, the framework can automatically output the best neural architecture. In other words, we just need to prepare the input and output data, and this framework can intelligently build a high-quality neural architecture for the given data.

In a neural network, each layer usually contains different parameters, which are critical information for the network transformation. According to Zoph and Le (2017), we specify the parameters of each layer in the form of a variable-length string. Subsequently, the proposed IAAS framework is applied to these strings for intelligent implementation of network transformation.

With the seq2seq technique (Bahdanau et al. 2015), each string containing the information of the neural architecture layer is first processed by an embedding layer in the IAAS framework, making it a fixed-length vector. We follow the technique given in Cai et al. (2018) and set the vector length as 16 in this study. Hence, if the network architecture has N layers, we then obtain a matrix with the size of $N \times 16$ to represent the architecture information. Similar to Cai et al. (2018), an encoder network built with a bidirectional LSTM network is subsequently used to learn the representations of this matrix. The fixed-length vector from the embedding layer is necessary for the encoder network. We present an illustrative example of the IAAS framework with a four-layer input architecture in Figure 1. It can be observed that after the encoder network, the learned features are simultaneously fed into three actors, namely wider actor, selector actor, and deeper actor, which are the cores of the network transformation control. Selector actor pioneers the decision-making process with four output action candidates, that is, “unchanged,” “pruning,” “widening,” and “deepening.” Briefly, a pruning action is to shrink the network with the movement pruning algorithm (Sanh et al. 2020), whereas a widening or deepening action is sent to the wider actor or deeper for the network enlargement. It can be noted that the selector actor automates the process of widening or deepening the network, successfully avoiding the manual settings for the network transformation in the EAS framework (Cai et al. 2018). After the network transformation operation, the updated architecture will be

Figure 1. Proposed IAAS Framework with an Example of a Four-Layer Input Architecture



entered into a net pool with a limited size. When the net pool receives all the transformed networks at one search episode, a heuristic screening algorithm is implemented to eliminate those networks with bad performance. The rest of the networks will then be implemented for the next-round transformation.

3.2. Markov Decision Process

In this research, the sequential optimization of network architecture is formulated as a Markov decision process (MDP), which is solved by reinforcement learning (RL). Specifically, the state s_u is the network architecture at the uth transformation step (we use u here because t has been used as the notation for the time series data). Three RL agents are defined to have the different action spaces so that they can adaptively select the appropriate transformation actions and intelligently conduct the specific network transformation to update the network architecture (see detailed information in Section 3.3). We evaluate the reward signal r_u with the model forecasting performance on the testing data sets. In particular, we define r_u as

$$r_u = \frac{1}{RMSE_u}, \tag{7}$$

where $RMSE_u$ indicates the rooted mean squared error of forecasting accuracy at the uth transformation step. This new reward function with $RMSE$ can award the agent with a nonlinear rate so that the same magnitude reduction in a smaller $RMSE$ is more prominent than that in a higher $RMSE$. The goals of the three RL agents are all to maximize the discounted reward, that is, $R_u = \sum_{i\geq 0}\gamma^i r_{u+i}$, in expectation, where $\gamma \in [0,1)$ is the discount factor.

At the transformation step u, the three agents work together to make a transformation decision a_u based on the parameters θ_u and the current network structure s_u. The policy $\mu(\cdot, s_u)$ generated by the agents is also recorded as the behavior policy to train the actor in the present and future steps. After the transformation, a new network structure s_{u+1} is obtained. With a few training iterations, the forecasting accuracy is evaluated and the reward signal r_u is calculated. This provides the data $(s_u, a_u, r_u, \mu(\cdot|s_u))$ for the RL algorithm to update the parameters θ_u. After updating θ_u, a new set of parameters θ_{u+1} is achieved for decision making at the next step. This process is repeated until the search budget is exhausted.

We aim to optimize the nondifferentiable expected long-term reward $\mathbb{E}(R_u)$ by using a policy gradient method called actor-critic with experience replay (ACER) (Wang et al. 2017). This algorithm is used to train three agents simultaneously, allowing us to update each agent with the most recent information based on the performance of all agents. For example, the selector agent will utilize the transformation actions taken from wider and deeper agents to prioritize its decision (widening, deepening, pruning, or keeping network unchanged). Without such a design, the selector agent may not well evaluate the decisions made before, leading to an erratic decision possibly to transform the networks. After each step of the network transformation, a new sampling trajectory $\{s_0, a_0, r_0, \mu(\cdot s_0), \ldots, s_u, a_u, r_u, \mu(\cdot s_u)\}$ is generated. ACER is then used to calculate the policy gradient and update the parameters θ_u. Moreover, by transforming the network structures in net pool, multiple sampling trajectories are generated and used for parameter updating. Each of these trajectories is composed of a series of decisions. With these trajectories, we can optimize the parameters of actor and critic networks, leading to searching a quality network structure because the reward signal is based on the network's performance. Please refer to the detailed information of how to use the ACER algorithm to update the parameters of RL agents in Online Appendix D and how RL is involved in the IAAS framework in Online Appendix E.

3.3. Actor Networks

In this section, we propose three actor networks, namely selector actor, wider actor, and deeper actor. Corresponding to each actor network, we construct a critic network with two FCN layers. The actor network is used to output actions, whereas the critic network is exploited to evaluate the goodness of the actions. In the following, we introduce detailed information about the three actor networks.

3.3.1. Selector Actor. The architecture of the selector actor is shown in Figure 2. As seen, the embedded representation of the current network architecture is fed into the bi-LSTM encoder network for feature extraction. In bi-LSTM, the features are extracted from both the forward and backward information flows of the input. In our problem, the forward and backward information flows to capture the dependencies among different layers in the current network architecture. The objective of having the selector actor is to determine the specific transformation operation for the current model architecture. Thus, only the features at the end of these two information flows are fed into the selector actor (we mark them as filled in Figure 2) because they contain the accumulative information and represent the overall characteristics of the current architecture. Then, the output of the selector actor indicates the probabilities of four actions, that is, keep the network unchanged, prune the network, widen the network, and deepen the network. In Figure 2, they are denoted as P(unchanged), P(prune), P(wider), and P(deeper), respectively. Instead of selecting the action with the highest probability, we leverage a distribution sampling method. Specifically, we use the softmax function to normalize these four probabilities (x_i, $i = 1,2,3,4$):

$$\sigma(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}, i = 1, 2, 3, 4. \tag{8}$$

We then obtain a discrete probability distribution with these normalized probabilities. An action is sampled from this discrete distribution as the final action of the selector actor. The distribution sampling method can both ensure the selector actor exploits the learned experience (preferably select the action with a higher probability) and

Figure 2. (Color online) Selector Actor Net (P, Probability)

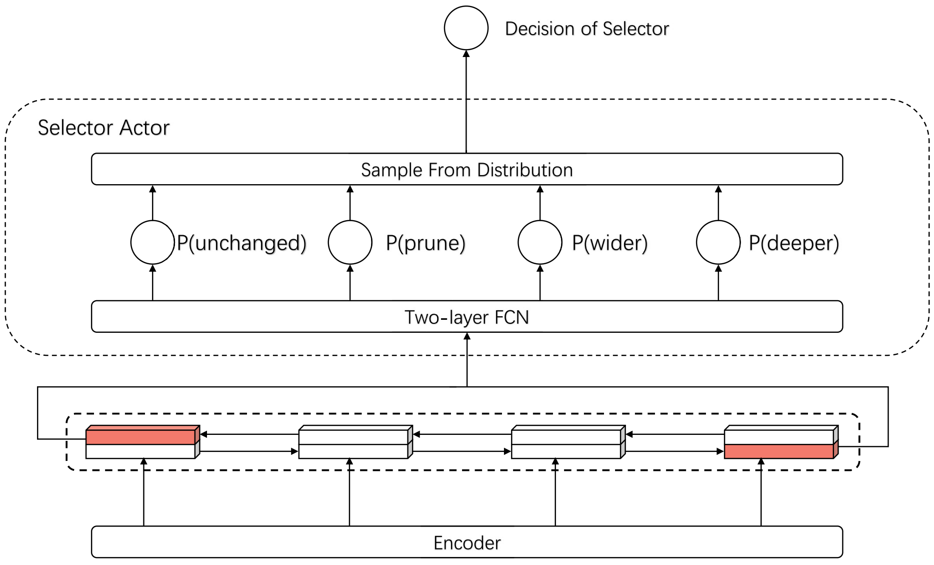
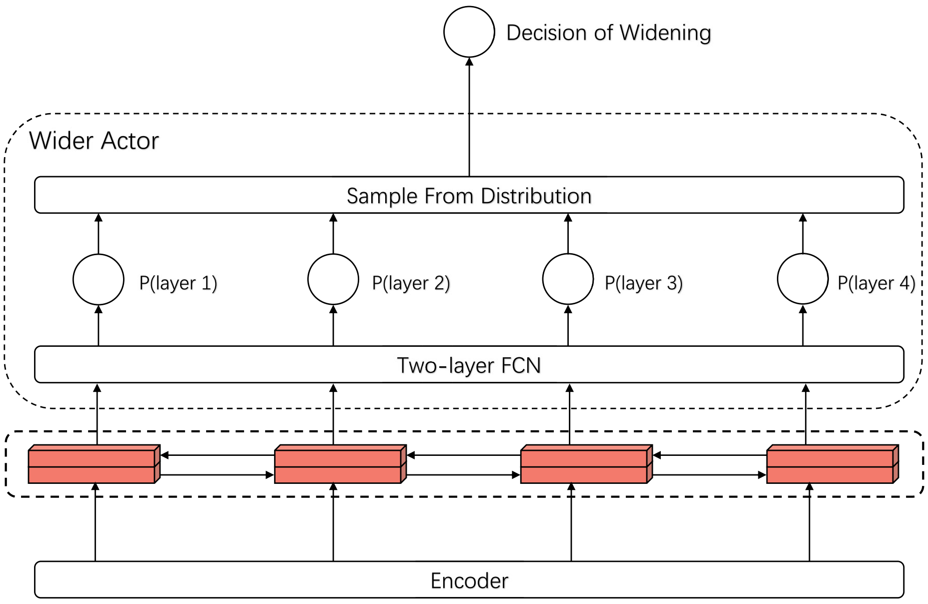


Figure 3. (Color online) Wider Actor Net with an Input Neural Architecture with Four Layers



Notes. All four layers have the same type of neural structure in this example. P, probability.

maintain the ability to explore new strategies (still possibly select the action with a lower probability). Thus, it can prevent the selector actor from trapping into the local optimum. Given the advantage of the distribution sampling method, we also include it in the design of the wider actor and the deeper actor.

For each transformation step, we use the current selector actor to generate the target policy π_s (see Online Appendix D) and exploit the policy gradient in Equations (D.4) and (D.5) to optimize the parameters in selector actor network and critic network with Adam optimizer. For the wider and deeper actors, we follow a similar procedure to optimize the parameters of the actor and critic networks.

3.3.2. Wider Actor.

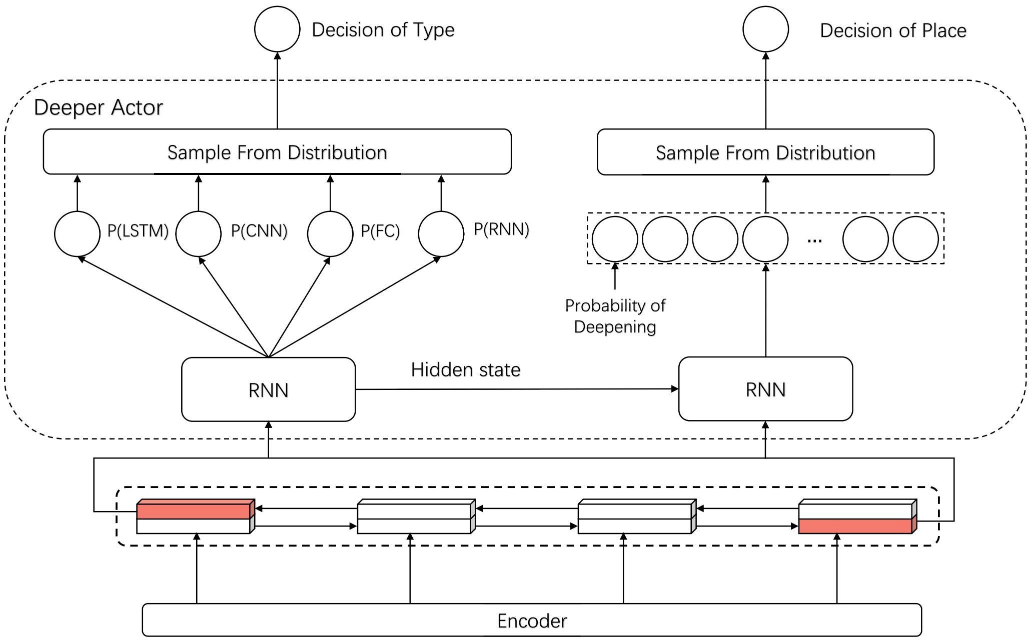
This actor is designed to widen a network layer with more units. Figure 3 shows the structure of the wider actor net. Similar to the selector actor, we also use an FCN to learn the features from the encoder network. Different from the selector actor, the wider actor aims to evaluate all layers and increase the units for a specific layer. Thus, the layer-wise features are fed into the FCN to determine actions (We mark these features as filled in Figure 3). It is worth noting that a shared FCN is used to generate the probability of widening actions for all layers. Thus, the FCN still remains the same after changing the number of layers (i.e., after the deeper transformations). The wider actor net will output four probabilities for a four-layer neural architecture as depicted in Figure 3. Instead of directly widening the layer with the highest probability, we normalize these probabilities and make the normalized probability as a discrete probability distribution, which is the same as that in the selector actor net. With such a distribution, we can sample an action to evaluate which layer should be widened eventually.

After selecting the layer to be widened, the wider actor needs to specify the number of units after the widening operation. The units for different network structures represent different items. For example, in FCN, the units represent the neurons, whereas in CNN, the units represent the channels. We set a widening rule with a self-defined sequence, for example, $[4, 8, 16, 32, \ldots, K, K+16]$. It denotes that if the current layer has four units, we then widen it to eight units; if the current layer has 8 units, we then widen it to 16 units; if the current layer has K units with $K \geq 16$, we then widen it to $K+16$ units. Hence, from a theoretical perspective, K could be infinitely large. If the number of units in the widening layer is not the exact number in the sequence, we will first widen the number to the nearest one. For instance, we widen a CNN layer with three output channels; then, we should first widen it to a CNN layer with four output channels. Therefore, this setting is quite accommodating because we do not need to manually set the maximum widening units.

3.3.3. Deeper Actor.

This section proposes a powerful deeper actor to deepen the given network. Specifically, we have four-layer candidates, namely, FCN layer, CNN layer, RNN layer, and LSTM layer. In the process of actor training, we believe any layer of the given network architecture can be deepened. For comparison, the Net2Deeper

Figure 4. (Color online) Deeper Actor Net



Note. P, probability of using CNN, FCN, or RNN layers.

actor in the EAS framework set a limitation that only an FCN layer can be added on top of all CNN and pooling layers for a given network. Therefore, to make the deepening process more flexible, the deeper actor is designed with the structure presented in Figure 4. Similar to the selector actor, the learned last two features (we mark them as filled in Figure 4) from the encoder network are selected because they represent the characteristics of the current model architecture. These features are fed into an RNN structure with a softmax activation function. There is a two-step action for the deeper actor. The first step is to determine what type of layer (e.g., FCN, CNN, RNN, or LSTM layer) will be added, which is also determined by the distribution sampling method introduced previously. The second step is to select the position of the newly added layer using the information from the first step and learned features from the encoder network. Briefly, we index each layer in the given network and calculate the probability of inserting the newly added layer. Then, the position to insert can be selected using the distribution sampling method, and the model can be accordingly deepened. As to the output unit amount of the inserted student network layer, we have the following rule. For an inserted student CNN layer, the output channel amount should equal that of its teacher network layer. We set the kernel size in CNN to three because the functionality of a kernel with any size can be equally represented by such a kernel size (Simonyan and Zisserman 2015). Meanwhile, we set the stride to one when the CNN layer is selected in the first step. Moreover, for an inserted student RNN or LSTM layer, the output unit amount should also be equal to that of its teacher network layer. However, for an inserted student FCN layer, the output unit amount should be equal to the time series length of the output data for its teacher network layer. For instance, if inserting an FCN layer to the network and its teacher network layer is a CNN layer with an output data size of 168×3, the output unit amount for this student FCN layer is 168. Conversely, if inserting an RNN layer to the network and its teacher network is this same CNN layer, the output unit amount for this student RNN layer is three.

3.3.4. Pruning Implementation. This section describes how the pruning action is implemented. Unlike the wider and deeper actions determined by RL, the pruning action is deterministic, derived from the movement pruning method (Sanh et al. 2020). As introduced in Section 2.3, we update the importance scores in each layer along with the corresponding weights. In FCN, RNN, and LSTM, one output neuron is a unit, whereas in CNN, one output channel is a unit. We create an importance score $\boldsymbol{\lambda}^{(i+1)}$ for each output unit in layer i so that the importance scores of each layer can be represented as a vector that has the same size as the output units. When the selector actor takes the pruning action, we first concatenate the vectors of importance scores from each layer to generate a unified vector $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}^{(2)}, \ldots, \boldsymbol{\lambda}^{(n+1)}\}$ (assuming that we have n layers in total). After that, we use Equation (4) to obtain the pruning mask $\boldsymbol{m}$, which selects the top $v\%$ important units by setting the importance mask value to one and zero for the rest. In our study, this value is set to $v\% = 90\%$. Once the mask $\boldsymbol{m}$ is determined, we break it into n vectors and each vector represents the mask values of a layer. Hence, we will obtain the mask values for all units in the network. Subsequently, for each layer, we perform the pruning action described in Section 2.3 to discard the units whose mask values are zero.

3.4. Net Pool

A net pool is developed in our proposed IAAS framework to iteratively remove the transformed neural architectures with bad performance to improve the quality of the searched architectures. This net pool has a heuristic screening algorithm that is similar to the standard genetic algorithm (GA). In the following, we present the process of how the net pool works. Initially, suppose we have S_Q network structures in the net pool and transform each of them with the proposed framework. After that, all the transformed networks are added to the net pool, and there are $2S_Q$ network structures in the pool. Next, other M networks are randomly generated and added to the pool. This random generation aims to reduce the possibility of premature phenomena in the results. All the $2S_Q + M$ networks are evaluated based on their forecasting performance. Moreover, the capacity of our net pool is denoted as C_Q (the maximum number of network architectures in the net pool), which means $C_Q \geq S_Q$ or M. We require the initial settings for S_Q and M as $2S_Q + M \geq C_Q$. Therefore, after ranking their forecasting performance from the best to the worst, we only keep the top C_Q network architectures in the net pool. Starting from this search episode, S_Q is equal to C_Q. The search process usually needs to take hundreds of episodes. After the search process is finished, we find the best network in the pool as our searched result. It should be noted that the proposed IAAS framework can be regarded as a two-stage operation in each search episode. Briefly, at the first stage, the RL-based network transformation will be implemented to transform each network structure in the net pool. At the second stage, the transformed networks will be re-entered into the net pool and a heuristic algorithm will be implemented to kick off the network structures with bad performance. In the EAS framework (Cai et al. 2018), it does not have such a net pool component, and the transformed network will be automatically used for the next-round transformation. This is like using one-fixed length trajectory for RL-based meta-controllers to collect samples. For a large search space, one trajectory usually is not enough. As a comparison, this net pool provides multiple candidates for RL-based meta-controllers to explore the search space. To better describe this heuristic screening algorithm, we present it in Algorithm 1 in Online Appendix E.

The elimination of networks in this heuristic approach happens under two scenarios. In the first one, if the performance of the transformed network $\mathcal{A}'$ is worse than that of the others (all the transformed networks and their parent networks) in the net pool, this network will be eliminated. In the second one, if the performance of both the network $\mathcal{A}$ and its transformed network $\mathcal{A}'$ have worse performance than the others, these two will be eliminated from the net pool, resulting in the termination of this network transformation trajectory. For the first scenario, the network $\mathcal{A}$ will be used for the next-round transformation, and its transformation trajectory will continue. For the second scenario, the transformation trajectory of the network $\mathcal{A}$ will be truncated. As noted, if the computational budget is fixed, the trajectory truncation has been demonstrated in Poiani et al. (2023, 2024) as an effective method to improve the performance of the RL agent, as this will allocate more computational budget to collect samples for the RL agent at the early stage of interaction with the environment. Note that the computational budget indicates the total number of action times in Poiani et al. (2023). For our IAAS, the computational budget is fixed (e.g., 1,000 action times), whereas the number of trajectories and the length of each trajectory are not fixed. The computational budget is equal to the sum of the lengths of all the trajectories. Moreover, randomly generating one network structure at each iteration is also like providing more early-stage samples to the RL agent. This is because each randomly generated network is the start of a new MDP trajectory.

Notably, the initial net pool is constructed from all network types such as FCN, CNN, RNN, and LSTM with one layer that has four units. In the screening process, we first identify the maximum layer number $(\tilde{A})$ and the maximum unit number of one layer $(\tilde{B})$ in the existing networks. Then, we use $\tilde{A}$ and $\tilde{B}$ as the upper bounds to randomly generate M networks as mentioned previously. In Algorithm 1, the stopping criterion is used to stop the architecture search process. Hence, if the stopping criterion is not satisfied, the net pool will be dynamically updated. This stopping criterion is problem dependent. In this work, this criterion is set to a fixed number of search episodes (e.g., 200) to validate the performance of our search scheme.

4. Numerical Results

In this section, we present our results by conducting numerical experiments using the proposed IAAS framework to develop electricity forecasting models in power systems. Our experiments include two streams of forecasting: load forecasting from the electricity consumption side and wind power forecasting from the electricity production side. In the following, we first describe the data information of these two main experiments. Subsequently, we present the experimental settings for the proposed IAAS framework and baseline methods. Next, we demonstrate and discuss the experimental results to validate the advantages of the proposed IAAS framework. Finally, we conduct an ablation study to demonstrate the importance of incorporating the selector actor and net pool components in this framework.

4.1. Data Description

We perform our experiments based on the time series data to build forecasting models of electricity load demand and wind power generation in power systems. For the electricity load forecasting experiment, two publicly available data sets of Maine (ME) and New Hampshire (NH) for the year 2020 are collected from ISO New England Inc. Similarly, for the wind power forecasting experiment, we use two wind farm (WF) data sets from a Chinese wind power company. Because the electricity time series data are typically influenced by various seasons (Jalali et al. 2021a, Qin et al. 2021), we classify each data set depending on the season, that is, spring, summer, autumn, and winter, respectively. Consequently, there are sixteen independent subsets for our experimental evaluations. The detailed data description can be found in Online Appendix F.

4.2. Experimental Settings and Performance Metrics

Comparison experiments are imperative for the demonstration of forecasting model performance. In this study, we exploit three classical ML methods, that is, support vector regression (SVR), random forest (RF), and ridge regression (RR), as baselines. Moreover, we use the following seven DL models to showcase the advantages of our proposed IAAS framework. First, we make use of three traditional DL methods, which involve the CNN, LSTM network, and CNN+LSTM network. Second, we use three proposed networks in the literature, where ResNet and ResNetPlus were derived from Chen et al. (2018) and deep adaptive input normalization (DAIN) was introduced by Passalis et al. (2019). Briefly, Chen et al. (2018) proposed two residual neural networks, where ResNetPlus was developed based on ResNet by using a side block technique; Passalis et al. (2019) introduced the DAIN method for the time series data to build a better forecasting model. Third, we use the latest SNAS framework (Chen et al. 2021), which targets the MTF model development with the NAS technique. We put the details of experiment settings in Online Appendix G. Last, after conducting the trial-and-error simulation, for IAAS, we determine to use 256 as the batch size, 200 as the maximum search episodes, and 50 epochs for each search episode. The number of search episodes directly has an effect on the search quality of network structures. Therefore, before the following experiments, we conduct a sensitivity analysis to investigate whether the use of 200 episodes is reasonable. The results are presented in Online Appendix H.

Based on the existing literature in both load forecasting and wind power forecasting (Chen et al. 2018, 2023; Çevik et al. 2019), we use two evaluation metrics, namely RMSE and MAE, to quantify the performance of forecasting results:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}, \tag{9}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|, \tag{10}$$

where Y is the actual value, $\hat{Y}$ is the predicted value, and N is the number of forecasting time points.

4.3. Results and Analysis

4.3.1. Electricity Load Demand Forecasting Experiments. Prior to the discussion of electricity load forecasting performance, we analyze the experimental results derived from our proposed IAAS framework regarding the network architecture. Given that each data set is split by four seasons, we thus obtain eight network structures of load forecasting models as presented in Table 1. To describe the architecture of each model, we leverage an example of the ME-autumn case for illustration. ME-autumn model is a three-layer network model: the first layer is a CNN layer with 15 output channels, and the second and third layers are both FCN layers with 14 and 24 output units, respectively. As observed, all these eight network structures are different from each other (the largest one has seven layers (see ME-winter case) and the smallest ones have two layers (see NH-spring case and NH-autumn case)). Because all the training data of these eight cases are different, the model structure differences indicate that the proposed IAAS framework is likely to automatically adapt to the given data set by generating an appropriate structure. It is worth noting that different network structures could generate similar performance. Observing

Table 1. Network Structures of Load Demand Forecasting Experiments Using IAAS Framework

Cases	Network structures
ME-spring	rnn-3→rnn-3→rnn-3→rnn-3→rnn-3→rnn-3→conv-24→fc-16
ME-summer	fc-3→fc-3→fc-3→fc-3→fc-3→conv-6→fc-5→fc-8→fc-8
ME-autumn	conv-15→fc-14→fc-24
ME-winter	conv-3→rnn-3→rnn-3→rnn-3→conv-3→rnn-3→rnn-3→rnn-3→rnn-1→rnn-1→rnn-1→fc-144
NH-spring	conv-3→fc-12
NH-summer	rnn-4→rnn-4→fc-16→rnn-16→rnn-16→rnn-16→rnn-16→fc-16→rnn-16
NH-autumn	conv-4→fc-16
NH-winter	fc-17→conv-64→fc-2→fc-7→conv-67→conv-5→fc-18→rnn-36→fc-64

Figure A.3 in the Online Appendix, the ME-spring data pattern is similar to the NH-spring data pattern; however, the network structures of the ME-spring case and NH-spring case are quite different. The reason could be that the internal mechanism of the IAAS framework cannot guarantee the globally optimal search due to the limitations of RL and heuristic screening algorithm.

Figure 5 depicts the network transformation procedures of the ME-autumn case using the IAAS framework. As shown, the final network structure contains one CNN layer and two FCN layers. As introduced in Section 3.4, we initialize the networks in net pool with one LSTM layer, one CNN layer, one RNN layer, and one FCN layer. Each layer has four units. Because we have a random network generation step as described in Section 3.4, this example indicates that, at a specific search episode, the randomly generated network (conv-16 → fc-24) performs better than all existing networks. Therefore, this randomly generated network is kept in the net pool. After six transformations with widening, deepening, and pruning, this network is transformed to the one with the structure of "conv-15→ fc-14 → fc-24", achieving the best forecasting performance in the searching process.

The forecasting accuracy results of our proposed IAAS framework and 10 baseline models or methods with two evaluation metrics are presented in Table 2. Moreover, for the load forecasting accuracy evaluation, we also make use of root mean squared logarithmic error (RMSLE) as another evaluation metric because an electricity company usually prefers to overshoot the electricity, and the undershooting may not be able to serve the customers well.

$$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(Y_i + 1) - \log(\hat{Y}_i + 1))^2}. \tag{11}$$

We name the acquired load forecasting models using the IAAS framework as IAAS_LoadNet. We mark the best of these models as boldface in each case. As we can observe, the obtained IAAS_LoadNet models outperform the baseline models in most cases. Specifically, in terms of RMSE and RMSLE, IAAS_LoadNet models perform the best in almost all cases, except in the case of NH-summer; as for MAE, IAAS_LoadNet models exhibit the best performance in six of eight cases. However, IAAS_LoadNet models have close forecasting accuracies compared with the best models in the cases of ME-summer and NH-summer. Furthermore, we present the average forecasting accuracy of eight cases at the bottom row of Table 2 and determine that the IAAS_LoadNet achieves much better performance than all the baseline models on average. In detail, the RMSE, MAE, and RMSLE for the IAAS_load-Net model are 76.622, 60.758, and 0.058, respectively, and they are higher than the second-best model with 12.1% (RMSE), 11.2% (MAE), and 12.1% (RMSLE), individually. In addition, we plot the predicted and actual loads for

Figure 5. (Color online) Network Transformation Illustration (ME-Autumn Case)

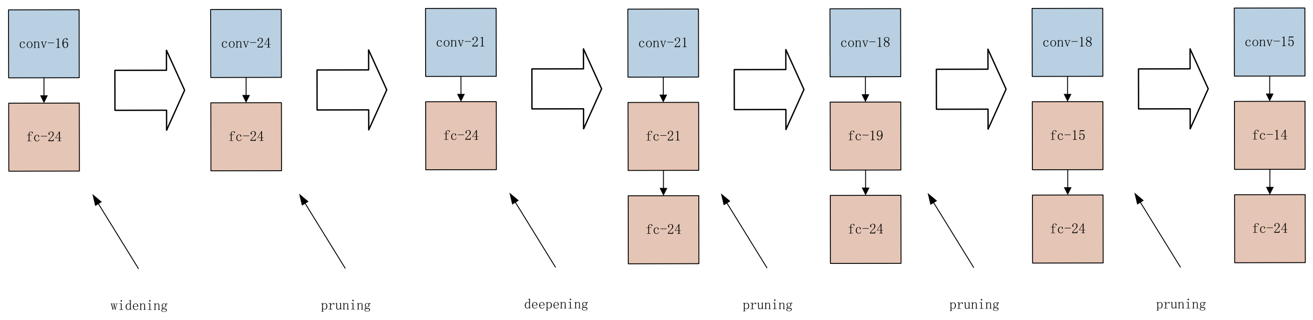


Table 2. Forecasting Performance of Electricity Load Forecasting Experiments

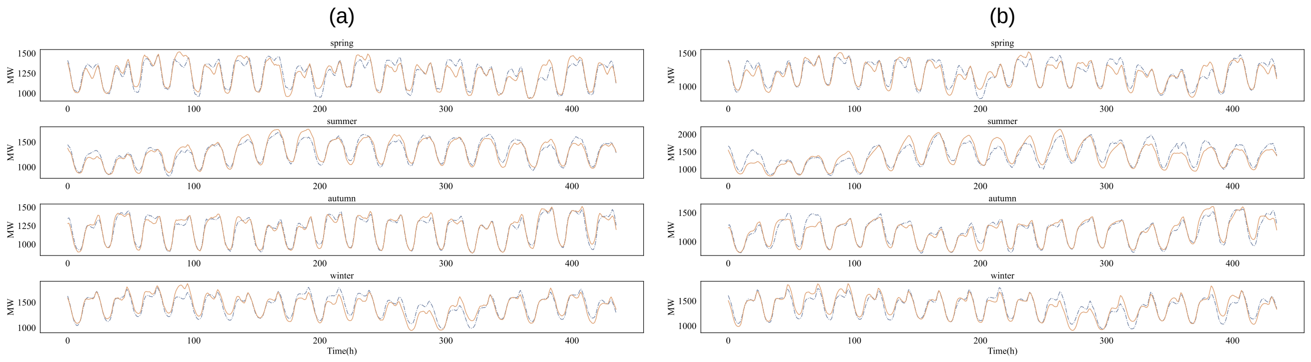
Case	Metrices	CNN-LSTM	CNN	LSTM	SVR	RF	RR	ResNet	ResNetPlus	DAIN	SNAS	IAAS_LoadNet
ME-spring	RMSE	70.559	74.583	116.771	115.912	69.665	68.557	64.038	64.585	65.686	62.866	**61.807**
	MAE	54.903	59.998	98.872	97.220	58.396	54.187	51.105	51.228	53.023	50.918	**48.879**
	RMSLE	0.055	0.059	0.092	0.099	0.055	0.054	0.050	0.051	0.052	0.050	**0.049**
ME-summer	RMSE	152.562	153.304	168.969	203.196	152.306	120.076	75.901	75.110	83.615	85.378	**69.154**
	MAE	120.538	122.502	133.090	172.666	116.565	99.416	**56.900**	57.567	66.371	64.145	58.187
	RMSLE	0.112	0.115	0.124	0.163	0.110	0.092	0.055	0.056	0.063	0.062	**0.052**
ME-autumn	RMSE	64.949	90.953	111.229	156.521	76.274	64.901	49.906	48.775	55.913	59.123	**38.483**
	MAE	49.159	76.833	91.189	124.190	59.291	49.470	39.064	37.165	43.572	49.721	**30.526**
	RMSLE	0.051	0.076	0.087	0.139	0.061	0.051	0.040	0.038	0.045	0.050	**0.031**
ME-winter	RMSE	91.504	107.586	129.708	150.629	111.414	117.403	89.133	98.878	91.950	106.098	**85.386**
	MAE	75.127	83.435	102.449	125.602	92.020	94.374	74.332	80.917	70.789	88.392	**68.773**
	RMSLE	0.067	0.078	0.094	0.110	0.081	0.087	0.065	0.073	0.067	0.078	**0.062**
NH-spring	RMSE	80.928	94.490	110.839	133.012	82.394	81.885	75.690	76.451	77.602	86.761	**68.749**
	MAE	65.067	76.260	85.261	110.609	65.379	63.703	61.054	60.529	63.256	69.859	**53.759**
	RMSLE	0.067	0.079	0.089	0.119	0.066	0.067	0.062	0.064	0.063	0.074	**0.058**
NH-summer	RMSE	170.803	191.243	198.392	263.721	196.841	**139.446**	154.906	150.601	161.118	167.340	144.016
	MAE	134.767	155.533	149.884	209.686	143.986	**105.739**	117.775	114.732	131.704	133.334	114.45
	RMSLE	0.120	0.134	0.136	0.189	0.134	**0.093**	0.106	0.102	0.113	0.111	0.096
NH-autumn	RMSE	106.226	133.442	130.408	207.988	110.012	98.570	94.535	86.676	84.303	78.102	**62.468**
	MAE	87.791	106.388	107.390	165.998	78.383	75.753	69.169	65.660	64.958	62.730	**45.985**
	RMSLE	0.088	0.113	0.104	0.186	0.086	0.079	0.076	0.071	0.069	0.068	**0.052**
NH-winter	RMSE	84.863	209.698	112.714	135.381	92.193	97.517	95.369	96.150	88.595	103.085	**82.911**
	MAE	69.722	184.262	91.061	108.405	74.449	78.587	78.075	80.440	72.134	83.888	**65.506**
	RMSLE	0.064	0.157	0.085	0.097	0.068	0.075	0.073	0.073	0.066	0.080	**0.062**
Average	RMSE	102.799	131.912	134.879	170.795	111.387	98.544	87.435	87.153	88.598	93.594	**76.622**
	MAE	82.134	108.151	107.400	139.297	86.059	77.654	68.434	68.530	70.726	75.373	**60.758**
	RMSLE	0.078	0.101	0.101	0.138	0.083	0.075	0.066	0.066	0.067	0.072	**0.058**

Note. The best one is in bold.

the test data sets of both ME and NH cases in Figure 6. As witnessed, our proposed IAAS framework can match the actual loads in 24 h-ahead points very well.

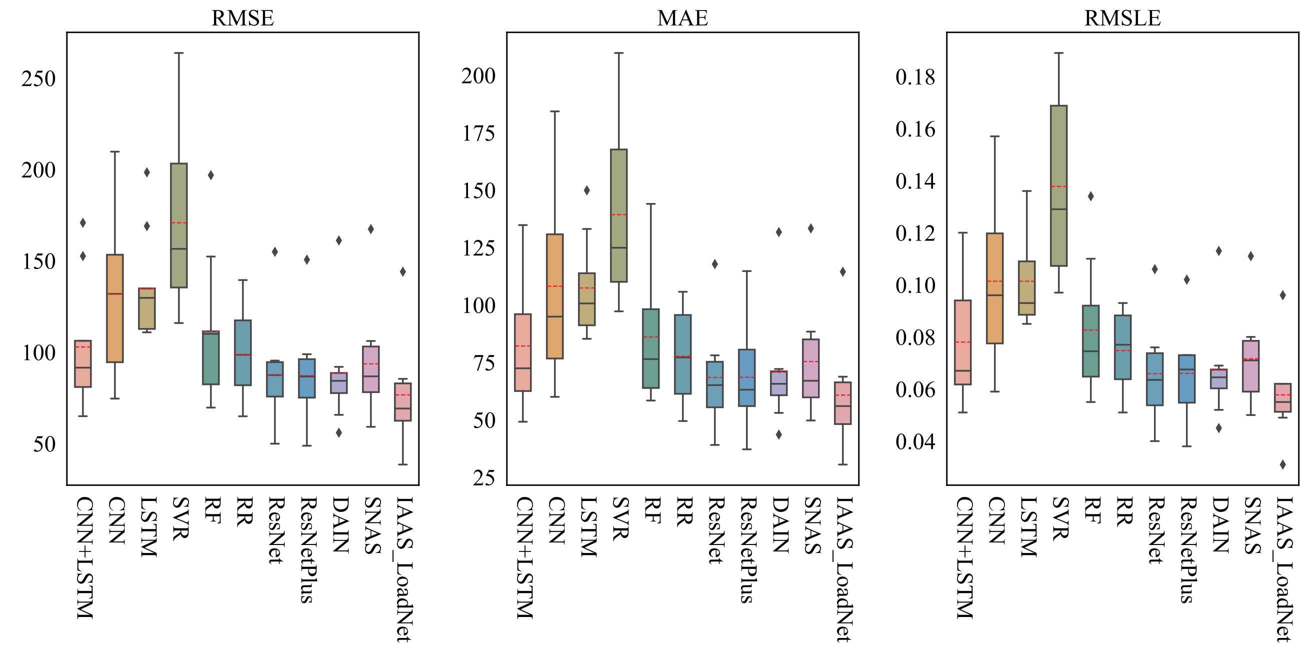
To further explore the stability of forecasting performance in the load forecasting experiments, we use the box-plot to demonstrate the variations in the forecasting accuracies, which are shown in Figure 7. As we can observe, all eleven models lead to various stabilities in terms of the three metrics; however, our proposed IAAS_LoadNet achieves competitively stable accuracies in terms of RMSE, MAE, and RMSLE (see the box body size). Such results also denote the necessity of using our IAAS framework to design a network architecture for various data sets. Moreover, we use the dashed line to indicate the average forecasting accuracy of each model in each box in terms of each metric. We expect the forecasting models to have the best average forecasting accuracy and low variance of forecasting accuracies. As seen, the IAAS framework can achieve this goal better than the baseline models. Based on this result and the forecasting accuracy results, we conclude that the proposed IAAS framework is able to adapt to the training data automatically for the development of load forecasting models.

Figure 6. (Color online) Comparison Results Between Predicted Electricity Load and Actual Electricity Load in Eight Testing Sets



Notes. (a) ME case. (b) NH case. The solid line represents the actual load values, and the dashed line represents the predicted values.

Figure 7. (Color online) Boxplot of Load Demand Forecasting Accuracies



4.3.2. Wind Power Forecasting Experiment. As described previously, we use two WF data sets considering the seasonal effects of conducting the wind power forecasting experiments, which constitutes eight cases in total. Table 3 presents the obtained network structures for each case after implementing our proposed IAAS framework. We name the obtained network models as IAAS_WindNet. As shown in the table, all the cases have different model structures. We present the forecasting accuracies of wind power with the two evaluation metrics in Table 4. It can be observed that the developed IAAS_WindNet models have shown the best performance in all cases in terms of RMSE and MAE. Averagely, the RMSE and MAE for the IAAS_WindNet model are 4.107 and 3.132, respectively, whereas those for the second-best model are 4.669 and 3.585, respectively. As presented in the table, our IAAS_WindNet models show better performance than the second-best models, with 12.0% (RMSE) and 12.6% (MAE), separately. We do not use RMSLE as the accuracy evaluation metric for wind power forecasting because RMSLE penalizes underprediction more than overprediction. In other words, when using RMSLE, generating more wind power than prediction will receive less penalty. On the electricity supply side, it pays more attention to the power system stability, which means producing more wind power than expected creates more challenges to the stability of the power system than producing less wind power than expected. Therefore, it is inappropriate to use RMSLE as the error measurement for wind power forecasting accuracy.

Consistent with the load forecasting experiments, we compare the predicted and actual wind power for both WF data sets in Figure 8 and find out IAAS_WindNet is likely to generate an accurate 24 h-ahead forecasting wind power. However, because of the more uncertain and intermittent nature of wind power compared with that of electricity load, sometimes it is challenging to accurately forecast the 24 h-ahead wind power generation as shown

Table 3. Network Structures of Wind Power Forecasting Experiments Using IAAS Framework

Cases	Network structure
WF1-spring	rnn-102→fc-102→rnn-102→fc-102→fc-102→fc-102→rnn-102→fc-102→fc-110→fc-108→fc-107→rnn-144→rnn-144→rnn-144→rnn-73→fc-4
WF1-summer	conv-64→fc-144
WF1-autumn	conv-8→fc-80
WF1-winter	fc-2→conv-8→rnn-13→fc-32
WF2-spring	conv-144→fc-108
WF2-summer	rnn-9→lstm-63→fc-48
WF2-autumn	lstm-80→lstm-80→conv-80→fc-144→conv-144→conv-144→conv-144→rnn-144→fc-144
WF2-winter	rnn-30→fc-1→conv-76→fc-2→conv-3→fc-11→lstm-118→rnn-29→fc-32

Table 4. Forecasting Accuracy Results of Wind Power Forecasting Experiments

Case	Metrices	CNN-LSTM	CNN	LSTM	SVR	RF	RR	ResNet	ResNetPlus	DAIN	SNAS	IAAS_WindNet
WF1-spring	RMSE	9.096	6.677	8.819	8.744	5.818	9.924	4.311	4.341	4.407	5.506	**3.901**
	MAE	7.967	5.239	5.949	7.539	4.478	7.945	3.417	3.366	3.412	4.472	**2.996**
WF1-summer	RMSE	4.951	4.154	4.168	4.358	5.646	4.807	3.816	3.842	3.615	3.956	**3.290**
	MAE	4.458	3.084	3.027	3.718	4.264	4.051	2.837	2.860	2.756	2.803	**2.614**
WF1-autumn	RMSE	8.263	4.301	8.247	8.603	4.051	4.161	3.113	3.539	2.918	4.296	**2.237**
	MAE	6.932	2.634	6.853	5.589	3.063	3.210	2.219	2.396	2.353	2.686	**1.633**
WF1-winter	RMSE	14.327	9.132	5.836	19.597	7.821	8.554	8.635	8.088	5.271	7.384	**4.500**
	MAE	13.189	7.509	4.862	18.450	6.707	7.063	6.050	5.974	3.995	6.094	**3.427**
WF2-spring	RMSE	13.656	11.862	6.836	13.758	5.561	7.283	6.146	5.948	5.600	7.661	**4.922**
	MAE	11.234	8.542	4.843	10.441	4.660	5.947	4.367	4.693	4.170	5.429	**3.953**
WF2-summer	RMSE	5.237	5.202	5.195	6.307	5.753	10.672	10.251	7.057	5.034	6.235	**4.870**
	MAE	4.275	4.171	4.145	4.460	4.251	6.630	7.093	4.941	4.043	4.402	**3.591**
WF2-autumn	RMSE	12.238	9.202	12.003	12.595	7.288	8.550	9.357	8.518	7.082	9.754	**6.031**
	MAE	9.706	6.988	9.681	9.364	5.917	6.628	7.314	6.585	5.405	7.419	**4.439**
WF2-winter	RMSE	23.142	11.559	5.038	21.512	5.005	9.044	4.551	3.374	3.424	8.570	**3.104**
	MAE	22.316	10.811	3.716	20.240	3.772	7.779	3.242	2.589	2.543	6.939	**2.402**
Average	RMSE	11.364	7.761	7.018	11.934	5.868	7.874	6.273	5.588	4.669	6.670	**4.107**
	MAE	10.010	6.122	5.385	9.975	4.639	6.157	4.567	4.176	3.585	5.031	**3.132**

Note. The best one is in bold.

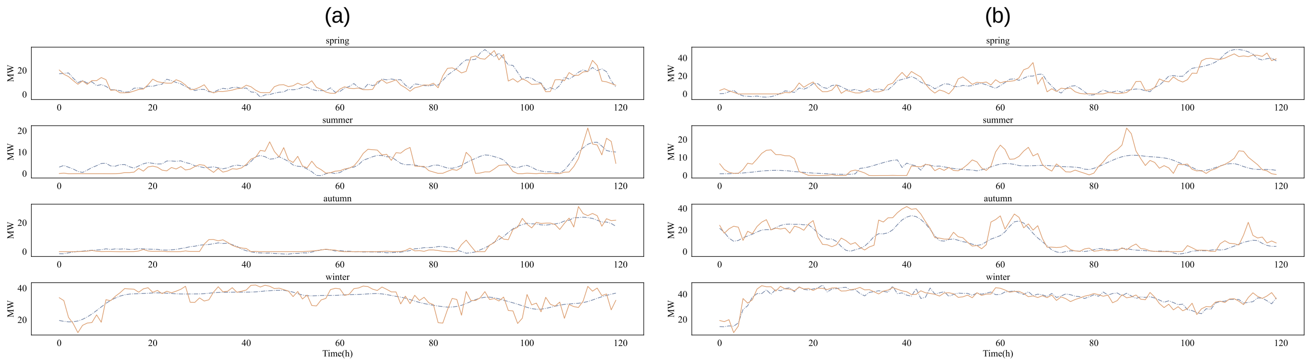
in Figure 8. We also draw the boxplot to evaluate the stability of the forecasting results in these two evaluation metrics in Figure 9. As shown in the figure, the IAAS_WindNet also achieves a competitively stable performance (see the box body size) compared with the others. The dashed line in the box indicates the IAAS models have the best performance on average than the others in terms of these two metrics. Hence, we conclude that the proposed IAAS framework can obtain better forecasting models for the 24 h-ahead wind power forecasting scenario compared with the existing models or methods.

4.3.3. Further Analysis.

As demonstrated previously, RL has achieved outstanding performance in network transformation control. However, the appropriateness of using RL in the IAAS framework requires further analysis. Therefore, we follow Cai et al. (2018) to make use of the random search algorithm to demonstrate the advantages of the selected RL algorithm from three perspectives, namely the forecasting accuracy in the search process, the average training time per epoch, and the number of average parameters of each obtained network. We put such a comparison analysis in Online Appendix I. The results indicate the selected RL algorithm is more efficient in achieving better RMSE forecasting accuracy. Moreover, the average training time per epoch of the selected RL algorithm is much smaller than that of the random search algorithm. Lastly, using the selected RL algorithm can result in a smaller number of the average parameters for each obtained model than using the random search algorithm. All in all, we believe it is appropriate to use the selected RL algorithm in the proposed IAAS framework.

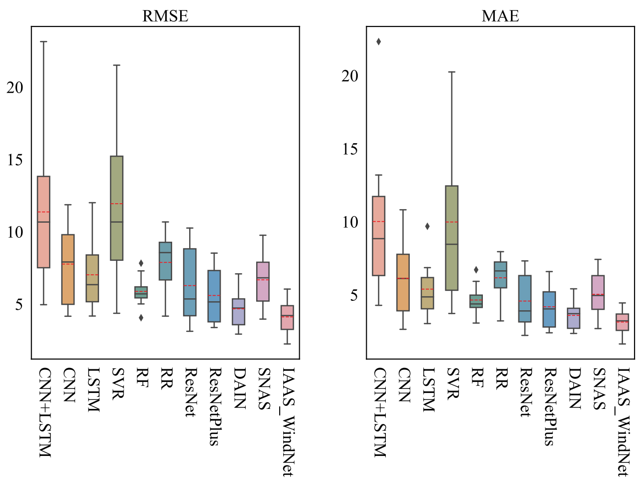
In addition, the load forecasting and wind power forecasting tasks in practice usually are based on the rolling-window framework but not set as previously shown. To better showcase the performance of the proposed IAAS

Figure 8. (Color online) Comparison Results Between Predicted Wind Power and Actual Wind Power in Eight Testing Sets



Notes. (a) WF1 case. (b) WF2 case. The solid line represents the actual wind power, and the dashed line represents the predicted wind power.

Figure 9. (Color online) Boxplot of Wind Power Forecasting Accuracies



framework practically, we conduct another experiment to compare the sequence-2-sequence (seq2seq) forecasting results in Online Appendix J between the IAAS framework and two existing transformer models from Vaswani et al. (2017) and Zhou et al. (2021a). The results demonstrate that under the seq2seq setting, the IAAS models still have better forecasting performance than the existing transformer models in terms of evaluation metrics used previously. Such results also indicate the broader impact of the proposed IAAS framework in real applications.

Last, we investigate how the heuristic mechanism works in the net pool. We develop three variants for demonstration. The first one is without the heuristic approach. Specifically, we initialize five networks in the net pool, transform each of them, and replace the networks in the net pool with these five transformed networks. Such a variant follows the scheme of the fixed-length transformation trajectory used in EAS (Cai et al. 2018). The second one is without randomly generating one network at each episode. The third one considers the trajectory truncation manner. Particularly, if there are 5 networks in net pool, there will be 10 networks after transformation. At this time, we randomly eliminate five networks and use the rest for further transformation. We place the results in Online Appendix K. As seen, our IAAS method generally achieves the better performance after comparing with each of these three variants. This result indicates that (1) this heuristic mechanism is effective in improving the quality of searching a neural architecture; (2) randomly generating one network at each iteration is important for RL agent; and (3) the network elimination strategy in the heuristic algorithm is likely to truncate the network transformation trajectory appropriately.

4.4. Ablation Studies

As discussed in Section 3, the primary differences between the IAAS and EAS frameworks are that the IAAS first uses a selector actor to intelligently determine how to transform the network instead of any manual process and second uses a network pool to improve the quality of the searched neural architectures. To demonstrate the advantages of these two novel features, we conduct ablation experiments in this section with three scenarios, which are based on three variants of the IAAS framework. The first variant is named IAAS_s, which indicates that the IAAS framework does not have the selector actor component. The second variant is named IAAS_n, which denotes the IAAS framework does not have the net pool component. The third variant is named IAAS_sn, which connotes that the IAAS framework does not have both selector actor and net pool components. It should be noted that for IAAS_s and IAAS_sn, we do not consider the manual setting that RL actors take four wider actions and five deeper actions as conducted in EAS (Cai et al. 2018) for each search episode. Instead, we consider randomly selecting an action of "widening," "deepening," "pruning," and "unchanged" for these two variants in each search episode. A random selection strategy might avoid a large network structure when the number of search episodes is large. Keeping the other hyperparameter settings the same, we obtain the forecasting accuracy results, which are presented in Tables A.4 and A.5 in Online Appendix L.

Based on Tables A.4 and A.5, we can conclude that (i) comparing IAAS to IAAS_s, selector agent generally can improve the model performance, which indicates the advantage of RL-based control in the selector agent; (ii) comparing IAAS to IAAS_n, net pool component can significantly improve the model performance, which demonstrates the benefits of enlarging the search space with different network candidates; and (iii) comparing IAAS to IAAS_sn, both of the selector and net pool component can jointly improve the model performance more than any of the other two variants. Such results denote the necessity of designing the selector and net pool components in IAAS framework.

5. Conclusion

Modern power systems require a real-time balance between electricity consumption and electricity production for their secure and smooth operation. To reduce the uncertainties and intermittencies from both the production and consumption sides, electricity forecasting has become one of the most effective methods. Thus far, most of the current electricity forecasting research studies have focused on applying DL techniques in constructing forecasting models. Although these forecasting models have demonstrated outstanding performance, they are developed primarily based on the designer's inherent knowledge and experience without explaining whether the proposed neural architectures are optimized or not. Moreover, these models cannot self-adjust to various data sets automatically. Although the NAS technique can automate the architecture search process by using optimization algorithms, most of the current techniques have computational issues, and only a few NAS techniques have been applied to the energy sector. Considering these, we propose an IAAS framework to search the high-quality neural architecture for the electricity forecasting model development.

The IAAS framework builds on and significantly extends the EAS framework (Cai et al. 2018). First, we develop the function-preversing transformation methods for RNN and LSTM considering their advantages in extracting features in time series data. Second, we incorporate pruning action into the IAAS framework so that our framework cannot only enlarge the network structures but also shrink them, which increases the search flexibility. Third, we consider a selector actor in IAAS so that we can intelligently and flexibly determine when to widen, deepen, prune the network or keep the network unchanged. Furthermore, we modify the wider and deeper actor networks more intelligently to transform the networks. Finally, we create a net pool component with a heuristic screening algorithm to expand the search space using multiple network structures as candidates to improve the probability of obtaining a high-quality network structure.

The experiments have been conducted on two types of electricity data, that is, load and wind power data. The experimental results demonstrate that the proposed IAAS framework predominately outperforms the 10 existing baseline models (or methods) according to the evaluation metrics of forecasting accuracy. Moreover, the proposed IAAS framework has competitively stable performance. Hence, we conclude that the proposed IAAS framework outperforms the existing methods. Moreover, we examine the appropriateness of the selected RL in the application of the IAAS framework from three perspectives by benchmarking with the random search algorithm. The results indicate the selected RL algorithm performs better than the random search algorithm in the forecasting accuracy, average training time per epoch, and average parameter number of each searched model. Furthermore, we make use of the seq2seq setting to compare the IAAS model with two existing transformer models. The results denote that the IAAS framework has a better performance, which means the IAAS framework is more suitable for practical applications in power systems. In addition, we implement an ablation study to showcase the significance of the selector actor and net pool components in the proposed IAAS framework after generating three variants of the framework. The ablation experiment results display that both the selector actor and the net pool component are likely to improve the forecasting model accuracy.

At the end of writing, we further discuss some extensions of the IAAS framework in Online Appendix M. We hope that our research is merely the first step toward more advanced analysis that focuses on electricity forecasting using the NAS technique, which could further improve the secured operation of the power systems. Conversely, our proposed framework can also be applied to other sequential or time series cases. Some examples include sales forecasting, travel demand forecasting, and speech recognition data to help improve their model accuracies. In addition, an extension of this research could focus on the theoretical analysis of the heuristic approach proposed in this study to explore why the integration of RL and this heuristic algorithm would improve the performance of the RL agent. Some computational experiments could be conducted to investigate whether this heuristic mechanism has a better performance than the other trajectory truncation methods.

Endnote

[1] Available at https://www.iso-ne.com/isoexpress/web/reports/pricing/-/tree/zone-info.

References

Agga A, Abbou A, Labbadi M, Houm Y (2021) Short-term self consumption PV plant power production forecasts based on hybrid CNN-LSTM, ConvLSTM models. *Renewable Energy* 177:101–112.

Ashok A, Rhinehart N, Beainy F, Kitani KM (2017) N2N learning: Network to network compression via policy gradient reinforcement learning. Preprint, submitted December 17, https://arxiv.org/abs/1709.06030.

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. Bengio Y, LeCun Y, eds. *Proc. 3rd Internat. Conf. Learn. Representations (San Diego)*.

Baker B, Gupta O, Naik N, Raskar R (2017) Designing neural network architectures using reinforcement learning. *Proc. 5th Internat. Conf. Learn. Representations* (OpenReview.net).

Baydin AG, Pearlmutter BA, Radul AA, Siskind JM (2018) Automatic differentiation in machine learning: A survey. *J. Machine Learn. Res.* 18:1–43.

Baymurzina D, Golikov E, Burtsev M (2022) A review of neural architecture search. *Neurocomputing* 474:82–93.

Bi X, Adomavicius G, Li W, Qu A (2022) Improving sales forecasting accuracy: A tensor factorization approach with demand awareness. *INFORMS J. Comput.* 34(3):1644–1660.

Billings BW, Smith PJ, Smith ST, Powell KM (2022) Industrial battery operation and utilization in the presence of electrical load uncertainty using Bayesian decision theory. *J. Energy Storage* 53:105054.

Cai H, Chen T, Zhang W, Yu Y, Wang J (2018) Efficient architecture search by network transformation. *Proc. 32 AAAI Conf. Artificial Intelligence*, vol. 32 (The MIT Press, Cambridge, MA), 2787–2794.

Çevik HH, Çunkaş M, Polat K (2019) A new multistage short-term wind power forecast model using decomposition and artificial intelligence methods. *Phys. A* 534:122177.

Chen T, Goodfellow IJ, Shlens J (2016) Net2Net: Accelerating learning via knowledge transfer. Bengio Y, LeCun Y, eds. *Proc. 4th Internat. Conf. Learn. Representations (San Juan, Puerto Rico)*.

Chen N, Qian Z, Nabney IT, Meng X (2013) Wind power forecasts using gaussian processes and numerical weather prediction. *IEEE Trans. Power Systems* 29(2):656–665.

Chen Y, Zhao J, Qin J, Li H, Zhang Z (2023) A novel pure data-selection framework for day-ahead wind power forecasting. *Fundamental Res.* 3(3):392–402.

Chen D, Chen L, Shang Z, Zhang Y, Wen B, Yang C (2021) Scale-aware neural architecture search for multivariate time series forecasting. Preprint, submitted December 14, https://arxiv.org/abs/2112.07459.

Chen K, Chen K, Wang Q, He Z, Hu J, He J (2018) Short-term load forecasting with deep residual networks. *IEEE Trans. Smart Grid* 10(4): 3943–3952.

Cheng AC, Lin CH, Juan DC, Wei W, Sun M (2020) Instanas: Instance-aware neural architecture search. *Proc. AAAI Conf. Artificial Intelligence*, vol. 34, no. 4 (The MIT Press, Cambridge, MA), 3577–3584.

Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: A survey. *J. Machine Learn. Res.* 20(1):1997–2017.

Gan L, Jiang P, Lev B, Zhou X (2020) Balancing of supply and demand of renewable energy power system: A review and bibliometric analysis. *Sustainable Futures* 2:100013.

Heo J, Song K, Han S, Lee DE (2021) Multi-channel convolutional neural network for integration of meteorological and geographical features in solar power forecasting. *Appl. Energy* 295:117083.

Hu Y, Hong Y (2022) Shedr: An end-to-end deep neural event detection and recommendation framework for hyperlocal news using social media. *INFORMS J. Comput.* 34(2):790–806.

Hu Q, Zhang S, Yu M, Xie Z (2015) Short-term wind speed or power forecasting with heteroscedastic support vector regression. *IEEE Trans. Sustainable Energy* 7(1):241–249.

Huang J, Pan K, Guan Y (2021) Multistage stochastic power generation scheduling co-optimizing energy and ancillary services. *INFORMS J. Comput.* 33(1):352–369.

Islam B, Baharudin Z, Raza MQ, Nallagownden P (2014) Optimization of neural network architecture using genetic algorithm for load forecasting. *Proc. 5th Internat. Conf. Intelligent Advanced Systems* (IEEE, Piscataway, NJ), 1–6.

Jalali SMJ, Ahmadian S, Khosravi A, Shafie-khah M, Nahavandi S, Catalão JP (2021a) A novel evolutionary-based deep convolutional neural network model for intelligent load forecasting. *IEEE Trans. Industrial Inform.* 17(12):8243–8253.

Jalali SMJ, Osório GJ, Ahmadian S, Lotfi M, Campos VM, Shafie-khah M, Khosravi A, et al. (2021b) New hybrid deep neural architectural search-based ensemble reinforcement learning strategy for wind power forecasting. *IEEE Trans. Industrial Appl.* 58(1):15–27.

Khodayar M, Kaynak O, Khodayar ME (2017) Rough deep neural architecture for short-term wind speed forecasting. *IEEE Trans. Industrial Inform.* 13(6):2770–2779.

Lahouar A, Slama JBH (2015) Day-ahead load forecast using random forest and expert input selection. *Energy Conversion Management* 103: 1040–1051.

Li Y, Huang J, Liu Y, Zhao T, Zhou Y, Zhao Y, Yuen C (2022) Day-ahead risk averse market clearing considering demand response with data-driven load uncertainty representation: A Singapore electricity market study. *Energy* 254:123923.

Liu D, Sun K (2019) Random forest solar power forecast based on classification optimization. *Energy* 187:115940.

Liu JJ, Hou Q, Liu ZA, Cheng MM (2022) Poolnet+: Exploring the potential of pooling for salient object detection. *IEEE Trans. Pattern Anal. Machine Intelligence* 45(1):887–904.

Liu Z, Sun M, Zhou T, Huang G, Darrell T (2018) Rethinking the value of network pruning. Preprint, submitted March 5, https://arxiv.org/abs/1810.05270.

Mendis HR, Kang CK, Hsiu PC (2021) Intermittent-aware neural architecture search. *ACM Trans. Embedded Comput. Systems* 20(5s):1–27.

Munawer ME (2018) Human health and environmental impacts of coal combustion and post-combustion wastes. *J. Sustainable Mining* 17(2):87–96.

Ng JYH, Hausknecht M, Vijayanarasimhan S, Vinyals O, Monga R, Toderici G (2015) Beyond short snippets: Deep networks for video classification. *Proc. IEEE Conf. Computer Vision Pattern Recognition* (Computer Vision Foundation/IEEE, New York), 4694–4702.

Nyashina G, Kuznetsov G, Strizhak P (2020) Effects of plant additives on the concentration of sulfur and nitrogen oxides in the combustion products of coal-water slurries containing petrochemicals. *Environment. Pollution* 258:113682.

Pan Z, Ke S, Yang X, Liang Y, Yu Y, Zhang J, Zheng Y (2021) AutoSTG: Neural architecture search for predictions of spatio-temporal graph. Leskovec J, Grobelnik M, Najork M, Tang J, Zia L, eds. *Proc. Web Conf.* (ACM, New York), 1846–1855.

Passalis N, Tefas A, Kanniainen J, Gabbouj M, Iosifidis A (2019) Deep adaptive input normalization for time series forecasting. *IEEE Trans. Neural Networks Learn. Systems* 31(9):3760–3765.

Pawlak Z (1982) Rough sets. *Internat. J. Comput. Inform. Sci.* 11(5):341–356.

Poiani R, Metelli AM, Restelli M (2023) Truncating trajectories in Monte Carlo reinforcement learning. Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, eds. *Proc. 40th Internat. Conf. Machine Learn.* (JMLR.org).

Poiani R, Nobili N, Metelli AM, Restelli M (2024) Truncating trajectories in Monte Carlo policy evaluation: An adaptive approach. *Advances in Neural Information Processing Systems*, vol. 36 (Curran Associates Inc., Red Hook, NY), 12141–12153.

Pryor SC, Barthelmie RJ, Bukovsky MS, Leung LR, Sakaguchi K (2020) Climate change impacts on wind power generation. *Nature Rev. Earth Environment* 1(12):627–643.

Qin J, Yang J, Chen Y, Ye Q, Li H (2021) Two-stage short-term wind power forecasting algorithm using different feature-learning models. *Fundamental Res.* 1(4):472–481.

Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, Le QV, et al. (2017) Large-scale evolution of image classifiers. Precup D, Teh Y-W, eds. *Proc. 34th Internat. Conf. Machine Learn.* (JMLR.org), 2902–2911.

Ren P, Xiao Y, Chang X, Huang PY, Li Z, Chen X, Wang X (2021) A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Survey* 54(4):1–34.

Sanh V, Wolf T, Rush A (2020) Movement pruning: Adaptive sparsity by fine-tuning. Larochelle H, Ranzato, M, Hadsell R, Balcan M-F, Lin H-T, eds. *Advances in Neural Information Processing Systems*, vol. 33 (Curran Associates Inc., Red Hook, NY), 20378–20389.

Shahid F, Zameer A, Muneeb M (2021) A novel genetic LSTM model for wind power forecast. *Energy* 223:120069.

Shepero M, Van Der Meer D, Munkhammar J, Widén J (2018) Residential probabilistic load forecasting: A method using gaussian process designed for electric load data. *Appl. Energy* 218:159–172.

Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. Bengio Y, LeCun Y, eds. *Proc. 3rd Internat. Conf. Learn. Representations (San Diego)*.

Solaun K, Cerdá E (2019) Climate change impacts on renewable energy generation. A review of quantitative projections. *Renewable Sustainable Energy Rev.* 116:109415.

Srivastava R, Tiwari A, Giri V (2019) Solar radiation forecasting using MARS, CART, M5, and random forest model: A case study for India. *Heliyon* 5(10):e02692.

Sunar N, Birge JR (2019) Strategic commitment to a production schedule with uncertain supply and demand: Renewable energy in day-ahead electricity markets. *Management Sci.* 65(2):714–734.

Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV (2019) MnasNet: Platform-aware neural architecture search for mobile. *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognition* (Computer Vision Foundation/IEEE, New York), 2820–2828.

Torres JF, Gutiérrez-Avilés D, Troncoso A, Martínez-Álvarez F (2019) Random hyper-parameter search-based deep neural network for power consumption forecasting. Rojas I, Joya G, Catala A, eds. *Proc. Internat. Work Conf. Artificial Neural Networks Adv. Comput. Intelligence* (Springer, Berlin, Heidelberg), 259–269.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, et al. (2017) Attention is all you need. von Luxburg U, Guyon I, Bengio S, Wallach H, Fergus R, eds. *Advances in Neural Information Processing Systems*, vol. 30 (Curran Associates Inc., Red Hook, NY), 5998–6008.

Wang Z, Bapst V, Heess N, Mnih V, Munos R, Kavukcuoglu K, de Freitas N (2017) Sample efficient actor-critic with experience replay. *Proc. Internat. Conf. Learn. Representations* (OpenReview.net).

Weron R (2014) Electricity price forecasting: A review of the state-of-the-art with a look into the future. *Internat. J. Forecasting* 30(4):1030–1081.

Xie L, Yuille A (2017) Genetic CNN. *Proc. IEEE Internat. Conf. Comput. Vision* (IEEE, Piscataway, NJ), 1379–1388.

Xiong B, Lou L, Meng X, Wang X, Ma H, Wang Z (2022) Short-term wind power forecasting based on attention mechanism and deep learning. *Electric Power Systems Res.* 206:107776.

Yang J, Jiang G, Wang Y, Chen Y (2023) An intelligent end-to-end NAS framework for electricity forecasting. http://dx.doi.org/10.1287/ijoc.2023.0034.cd, https://github.com/INFORMSJoC/2023.0034.

Zhang Z, Wei X, Zheng X, Li Q, Zeng DD (2022) Detecting product adoption intentions via multiview deep learning. *INFORMS J. Comput.* 34(1):541–556.

Zhou S, Li X, Chen Y, Chandrasekaran ST, Sanyal A (2021b) Temporal-coded deep spiking neural network with easy training and robust performance. *Proc. 35th AAAI Conf. Artificial Intelligence*, vol. 35 (The MIT Press, Cambridge, MA), 11143–11151.

Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021a) Informer: Beyond efficient transformer for long sequence time series forecasting. *Proc. Conf. AAAI Artificial Intelligence*, vol. 35 (The MIT Press, Cambridge, MA), 11106–11115.

Zoph B, Le QV (2017) Neural architecture search with reinforcement learning. *Proc. 5th Internat. Conf. Learn. Representations* (OpenReview.net).