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

Jin Yang, Yingying Huang, Guangxin Jiang, Ying Chen

School of Management, Harbin Institute of Technology, China

*Corresponding author: yingchen@hit.edu.cn

Abstract: Recent years have witnessed an exponential growth in developing deep learning (DL) models for the 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 the dynamically changing data patterns due to an inflexible design of their structures. Even though several latest studies have considered 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 quite time-consuming, computationally expensive and not intelligent, indicating that the NAS application in electricity forecasting area is still at an infancy phase. In this research study, we propose an intelligent automated architecture search (IAAS) framework for the development of time-series electricity forecasting models. The proposed framework contains two primary components, i.e., network function-preserving transformation operation and reinforcement learning (RL)-based network transformation control. In the first component, we introduce a theoretical function-preserving transformation of recurrent neural networks (RNN) to the literature for capturing the hidden temporal patterns within the time-series data. In the second component, we develop three RL-based transformation actors and a net pool to intelligently and effectively search a high-quality neural architecture. After conducting comprehensive experiments on two publicly-available electricity load datasets and two wind power datasets, we demonstrate that the proposed IAAS framework significantly outperforms the ten existing models or methods in terms of forecasting accuracy and stability. Finally, we perform an ablation experiment to showcase the importance of critical components of the proposed IAAS framework in improving the forecasting accuracy and reducing the structural complexity of the model.

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

1. Introduction

Electricity market has reshaped the electricity trade mode since the introduction of competitive market and deregulation processes during the early 1990s (Weron 2006). As electricity is a tradable commodity that cannot be stored on a large-scale, modern electricity markets necessitate a balance between electricity production and consumption in real-time (Lund 2005, Gan et al. 2020). Consequently, this requirement plays an integral role in keeping the stable operations and regulations of a power system (Huang et al. 2021). However, there are several factors that can affect this balance.
From the production perspective, many renewable energy sources—such as wind and solar—are increasingly contributing to the power grid systems with rapid growth (Solaun and Cerda 2019). According to the fuel report (International Energy Agency 2021), the overall global renewable electricity is predicted to be over 4800 GW in 2026, which is equivalent to the total power capacity of fossil fuels and nuclear combined in 2020. Compared to traditional energy resources, e.g., coal and gas, renewable energy is sustainable and clean (Meinshausen et al. 2009, Munawer 2018, Nyashina et al. 2020); however, its uncertain and intermittent nature brings significant challenges to the smooth and secured operation in power system (Chen et al. 2021a, Pryor et al. 2020). From the consumption perspective, electricity load can be correlated to various patterns related to industrial activities and weather conditions (see Chen et al. 2019, Jalali et al. 2021a). For instance, industrial and commercial consumers usually consume more electricity during the daytime than at night, and utilize more electricity during the summer than in spring or autumn. Nonetheless, these generic electricity usage patterns on any given day are still full of uncertainties (Metaxiotis et al. 2003) since most of the consumption behaviours are uncontrollable. To enhance the secured and reliable operation, electricity forecasting has become one of the most effective techniques to minimize such 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 the forecasting accuracy. As artificial intelligence (AI) techniques are gaining popularity rapidly, researchers have turned to utilizing advanced machine learning (ML) methods instead of traditional statistical time-series methods (e.g., autoregressive moving average (ARMA) and exponential smoothing) for the electricity forecasting model development. For example, Hu et al. (2016) used support vector machine (SVM) models to forecast wind power generation; Liu and Sun (2019) employed random forest (RF) models to predict solar power generation; Islam et al. (2014) exploited the artificial neural network (ANN) models to forecast load demand. Various other relevant research studies can be found in the literature, such as Chen et al. (2014), Lahouar and Slama (2015), Shepero et al. (2018), Srivastava et al. (2019), etc. On the other hand, remarkable success of deep learning (DL) can be witnessed in pattern recognition, object
detection, sales forecasting, and other applications (e.g., Bi et al. 2022, Hu and Hong 2022, Liu et al. 2022, Zhang et al. 2021, Zhou et al. 2021). Hence, we have witnessed a dramatic incline of 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 multi-channel convolutional neural network (CNN) model, and Agga et al. (2021) proposed two models: one is CNN+LSTM model and the other is convolutional LSTM model; in load forecasting, Chen et al. (2019) introduced two deep residual network models, and Jalali et al. (2021a) proposed an evolutionary-based deep CNN model. Compared to traditional ML methods such as SVM and RF, DL models like CNNs 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 self-adjust the deep neural network structures for adapting to various datasets, in order to produce high-quality forecasting models for power systems. There are two motivations behind our considerations. First, numerous prior research studies have designed the model structures based on their knowledge or experience (Elsken et al. 2019), especially in the energy sector. For example, Shahid et al. (2021) designed a three-layer LSTM model for wind power forecasting; Heo et al. (2021) developed a four-layer multi-channel CNN model for solar power forecasting; Jalali et al. (2021a) presented a three-layer CNN plus a fully connected network (FCN) model for load forecasting. Even though these research studies have conducted abundant experiments to demonstrate the advantages of their proposed models, they could not explain why these structures could provide higher forecasting accuracies. As noted, a larger (or deeper) network usually would achieve a better performance than a smaller network (Ng et al. 2015) without considering the over-fitting issue. In other words, it is still unknown whether a particular forecasting model could be further improved if using more CNN or LSTM layers. As described in Ren et al. (2022), designing an optimal neural architecture only based on inherent knowledge of human beings is problematic since it is difficult for people to jump out of their thinking paradigms. Second, the existing designed network structure is fixed for all scenarios as demonstrated
in the literature (e.g., Heo et al. 2021, Jalali et al. 2021a, Shahid et al. 2021), and cannot be self-adjusted based on new data. However, according to the above description, the power system can be influenced by climate conditions and industrial activities to create uncertainties and intermittencies in electricity production and consumption. Hence, a fixed model structure can hardly perform well in such diverse scenarios. If the data pattern is changed by some unknown factors, then the model trained with good performance on diverse datasets may encounter an unpredictable decline in forecasting accuracy. Henceforth, an automated neural architecture optimization algorithm is imperative for a power system to generate high-quality and self-adjusted forecasting models.

Technically, 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 presented on some public benchmark datasets. However, regarding electricity forecasting, only a few works have been reported, which implies that the development and application of the NAS technique in the power system is still at an infancy stage. Limited examples include that Khodayar et al. (2017) developed a NAS strategy based on 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 these 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. Even though several evolutionary-based search methods (e.g., Real et al. 2017, Xie and Yuille 2017, Jalali et al. 2021b) attempt to cover the whole architecture space, each candidate model in these studies has to be trained to convergence from scratch, resulting in a significant computational burden (Baymurzina et al. 2022). In addition to evolutionary methods, reinforcement learning (RL) is another paradigm that provides a more controlled search (e.g., Baker et al. 2017, Zoph and Le 2017). However, this strategy is also limited by computational issues, because each sampled architecture has to be trained from the beginning (Baymurzina et al., 2022).
In this work, we propose an intelligent automated architecture search (IAAS) framework, aiming to acquire the neural forecasting models for electric power systems efficiently. The proposed framework has two primary components: network transformation operation and control. Network transformation operation indicates how to transform a trained network as a starting point to obtain a new network. In other words, the new network does not need to be trained from scratch, which reduces the computational issues existing in the conventional RL-based search strategies. Network transformation control denotes where to widen or deepen the network intelligently to generate a network with diverse types of structures.

The key technical contributions of our work are to provide the theoretical function-preserving transformation of recurrent neural network (RNN) and the design of the IAAS framework. First, we harness the Net2Net transformation framework (Chen et al. 2016) to widen and deepen the CNN and FCN layers. However, energy data is in the form of time series, and only the spatial networks like CNN and FCN are not sufficient to capture the inherent temporal patterns. Therefore, we use RNN that are naturally suited to process the time-series or other sequential data (DiPietro and Hager 2020) in the Net2Net framework. Compared to spatial networks like CNN and FCN, RNN is more complicated since it contains the parameters to pass information through time. Hence, the transformation formulations for CNN and FCN in Chen et al. (2016) are difficult to extend to the RNN transformation development. We tackle these issues by reformulating the RNN operation and then deriving the RNN wider transformation and deeper transformation, respectively. To the best of our knowledge, this is the first study to conduct the RNN function-preserving transformation and apply it to the forecasting model construction in the energy area. Second, our IAAS framework has been developed based on the efficient architecture search (EAS) framework (Cai et al. 2018), which signifies the actions of RL-based meta-controller as transformations of the existing network. However, there are two main differences between IAAS and EAS frameworks. The first difference is that in EAS, the RL-based meta-controller is manually set in experiments to take five steps of deeper transformation and four steps of wider transformation at each search episode (Cai et al. 2018). However, this manual setting is not explained with a reason and can easily result in an over-parameterized network model. To address this issue, we propose an RL-based selector actor to intelligently determine whether to widen, deepen, or keep the
network unchanged at each search episode. With this decision, the network transformation can then be implemented. The second difference is that in EAS, after the network transformation, the new network is directly input into the next-round transformation. However, such a setting can easily cause the architecture search to drop into the local optimal state if the initial architecture exhibits good performance. To remedy this, we add a net pool with a constrained size as a buffer and propose a simple algorithm to manage the pool in order to keep a set of networks with good performance in it. Due to this innovation, the network entering into the transformation process might be a different one at each search episode, reducing the possibility in the generation of local optimal neural architectures. Besides these two primary differences, we also modify the RL-based network transformation meta-controllers in Cai et al. (2018) so that they are more intelligent in developing a neural architecture with mixed types of network structures.

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. Each case is evaluated by using two independent datasets. The wind power datasets have been acquired from a wind power company in China, while the load datasets have been collected from ISO New England Inc. The experiments contain two parts: comparison experiments and ablation experiments. In the comparison experiment, we use ten existing models or methods with standard performance metrics to demonstrate the effectiveness of our framework. As to the ablation experiments, we develop three variants of IAAS framework to showcase the significance of proposing selector actor and net pool components in the IAAS framework from two different perspectives – i.e., model forecasting accuracy and model structural complexity.

The remainder of this paper has been organized as follows: Section 2 gives the preliminaries of this study, including two classes of related works, background of RNN and description of feature-learning operation. Section 3 introduces the theoretical RNN network transformations. Section 4 proposes the IAAS framework and details the key components in it. Section 5 demonstrates the advantages of the proposed IAAS framework via comparison and ablation experiments. Finally, in Section 6, we discuss the experimental results and provide the concluding remarks.
2. Preliminaries

2.1. Related work

2.1.1. Neural architecture search with RL

As one of the search strategies in NAS, Reinforcement Learning (RL) has successfully obtained high-quality model architectures (see Zhong et al. 2018, Zoph et al. 2018). With RL, the generation of a neural architecture can be considered 'the agent's action. Moreover, the trained architecture performance on concealed data can be leveraged to estimate the search reward (Elsken et al. 2019). Hence, the agent receives a reward when the network is completely constructed. In literature, numerous RL-based approaches have been developed for NAS. For example, Zoph and Le (2017) started to use RNN as a controller to sample a sequential neural architecture and exploited REINFORCE policy to learn the controller. Another pioneering research work has been conducted by Baker et al. (2017), which applied a Q-learning algorithm with experience replay to NAS. As a comparison, the model accuracy in Baker et al. (2017) is lesser than that in Zoph et al. (2017) using the CIFAR-10 dataset (Krizhevsky and Hinton 2009), whereas the computational time required in the case of Zoph et al. (2017) is much higher than that in Baker et al. (2017). Nonetheless, both approaches require to train each sampled network architecture from scratch, resulting in much computational waste. To overcome this issue, many research works have proposed warm-started methods by inheriting the weights of the existing network. A representative example is the 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 one and contains two specific methodologies for CNN and FCN network transformations, namely $Net2WiderNet$ and $Net2DeeperNet$. Specifically, the $Net2WiderNet$ transformation approves to replace a model with an equivalent and wider model, while the $Net2DeeperNet$ transformation permits to replace a model with an equivalent and deeper model. Based on Chen et al. (2016), another work by Cai et al. (2018) proposed two RL-based meta-controllers, such as $Net2Wider$ and $Net2Deeper$ actors, to widen and deepen the trained network,
respectively. In these controllers, both the actors learn the architecture representations from an encoder built with a bidirectional LSTM network.

Notably, all these related research studies deal with the imaging data and have not incorporated the time-series data, especially in the energy sector. Compared to these extant approaches, our approach has several advantages. First, our proposed IAAS framework is an end-to-end framework that overcomes the manual settings for network transformations. Due to this feature, our proposed framework is probable to obtain a high-quality network model constructed with a small number of parameters. Second, our proposed IAAS framework considers the model architecture's local optimal state, which can help acquire a more accurate forecasting model. Third, our IAAS framework is suitable to operate with various types of data, including imaging and time-series data, since we bridge the knowledge gap by introducing the RNN function-preserving transformation to the literature. Fourth, our IAAS framework allows building a network with diverse types of structures, using CNN, RNN, and FCN, so that a powerful forecasting model can be produced.

2.1.2. Neural architecture search for multivariate time-series forecasting

As discussed above, energy data is characterized by time-series, and it has different features than that of grid data (e.g., images). Multivariate time-series (MTS) forecasting is a widely studied topic in literature. For MTS forecasting, the basic operation blocks should consider both intra-variable and inter-variable dependencies (Shih et al. 2019). Even though some latest deep learning research has studied MTS forecasting (e.g., Oreshkin et al. 2021, Wu et al. 2019 and Zheng et al. 2020), only a few studies have taken the NAS technique into consideration. Limited examples include a study by Pan et al. (2021), who designed a NAS method for predictions of Spatio-temporal graphs, which are vital structures to describe urban sensory data, e.g., traffic speed and air quality. However, this method is unsuitable for pure MTS forecasting since it leverages additional information to acquire the adjacency matrix required for the spatial convolution operation. Based on the work by Pan et al. (2021), Chen et al. (2021b) developed a scale-aware NAS (SNAS) method for MTS forecasting. Briefly, this method exploits cells that are directed acyclic graphs (DAGs) as building blocks to construct a network for each scale by cell stacking. The gradient-based optimization method is utilized in the NAS process.
As noted, this method attains a better forecasting performance than the other graph-based existing methods; however, it has two limitations in our viewpoints. First, this method requires a predetermined number of cells. This setting is dependent primarily on designers' experience and inherent knowledge. If the setting is not appropriate, it cannot achieve a high-quality structure, or it may take a lot of unnecessary time to reach a superior structure. Second, this method requires two stages that first obtain an optimized structure and then achieve the optimal weights based on the optimized structure. Rather than using an identical technique for NAS as discussed by Pan \textit{et al.} (2021) and Chen \textit{et al.} (2021b), our proposed framework is more flexible and efficient due to the following aspects. Specifically, our proposed framework does not limit the maximum number of architectures during the search process, which avoids the drawbacks of such a manual setting. Moreover, our framework is a one-stage search that can achieve the optimized structure with its corresponding optimal weights simultaneously.

2.2. Recurrent neural network (RNN)

Various research studies have preferred to use RNN or LSTM (a special case of RNN) (e.g., Dong \textit{et al.} 2018, Kisvari \textit{et al.} 2021, Yahya \textit{et al.} 2018) to build the electricity forecasting models since the connections between RNN nodes produce a directed or undirected graph along a temporal sequence (Samuel 2019). Compared to CNN and FCN, RNN not only employs a feed-forward structure to process the initial input to final output but also uses feedback loops to better process sequential and temporal data (Ahmed 2018).

To better display the operation of RNN, we utilize a simple inner structure as depicted in Fig.1. As seen, RNN layer $i$ consists of a $T$-time vector sequence of inputs $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_T^{(i)})$ and a $T$-time hidden vector sequence, $\mathbf{x}^{(i+1)} = (x_1^{(i+1)}, x_2^{(i+1)}, ..., x_T^{(i+1)})$. Considering the temporal information, besides the current inputs, the outputs from the last time period are other inputs for layer $i + 1$. $W^{(i)}$ and $H^{(i)}$ represent the corresponding weight matrices that connect inputs and outputs. It can be noted that the input at each time step, $x_t^{(i)}$, for $t = 1, 2, ..., T$, can be a scalar with one feature or a vector with many features. Observing in Fig. 1, the output generation $x_t^{(i+1)}$ depends on the information of both current inputs $x_t^{(i)}$ and outputs from the last time period $x_{t-1}^{(i+1)}$. Hence, the RNN process at time $t$ can be written as:

$$x_t^{(i+1)} = f_a \left( W^{(i)} x_t^{(i)} + H^{(i)} x_{t-1}^{(i+1)} \right),$$ (1)
where $f_{\alpha}$ is an activation function.

![Diagram of RNN layer](image)

Figure 1. The inner structure of an RNN layer

Note that the output must be updated at each time step, which is the primary difference between RNN and other spatial networks like CNN and FCN. When integrating RNN into our proposed IAAS framework, it is critically essential to conduct the function-preserving transformation for this structure. Even though Chen et al. (2016) introduced the network transformation methods for spatial networks such as CNN and FCN, it is tough to directly apply these methods to RNN due to the time-series characteristics in the outputs. To overcome such complexity, we describe a feature learning process in the spatial networks as follows and then derive the RNN reformulation from a feature-learning perspective given in Section 3.

### 2.3. Feature learning operation

This study divides the feature learning process into two phases: feature extraction and feature generation. Suppose that in an FCN, there are $p$ input features $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)})$ and $m$ output features $x^{(i+1)} = (x_1^{(i+1)}, x_2^{(i+1)}, \ldots, x_m^{(i+1)})$ for layer $i$, where the outputs are inputs for layer $i + 1$. The input weight matrix for layer $i$ is $W^{(i)} \in \mathbb{R}^{p \times m}$. The feature extraction is specified as a meta operation function $f_1 \left( x_j^{(i)}, W_{j,k}^{(i)} \right)$, for $j = 1, 2, \ldots, p$ and $k = 1, 2, \ldots, m$, which is a scalar multiplication. Subsequently, the feature extraction results $o_k^{(i)}$ for layer $i$ are obtained as follows:

$$o_k^{(i)} = \left( f_1 \left( x_1^{(i)}, W_{1,k}^{(i)} \right), f_1 \left( x_2^{(i)}, W_{2,k}^{(i)} \right), \ldots, f_1 \left( x_p^{(i)}, W_{p,k}^{(i)} \right) \right), \quad k = 1, \ldots, m. \quad (2)$$
The feature generation is denoted by an aggregating function that summarizes information from the extracted features, \( f_2 \left( o_k^{(i)} \right) = \sum_{j=1}^{p} f_1 \left( x_j^{(i)}, W_{j,k}^{(i)} \right) \), for \( k = 1, \ldots, m \). Then, we generate the output features \( x^{(i+1)} \), which are given by

\[
x^{(i+1)} = f_a \left( f_2 \left( o_1^{(i)} \right) \right), f_a \left( f_2 \left( o_2^{(i)} \right) \right), \ldots, f_a \left( f_2 \left( o_m^{(i)} \right) \right).
\]

(3)

Fig. 2 depicts a simple FCN example of such a feature learning process. As shown in the figure, layer \( i \) has two input and two output features. Hence, the meta operation \( f_1 \) is utilized to extract the knowledge from the input features (i.e., Extraction) and the aggregation operation \( f_2 \) is employed to accumulate the learned knowledge to produce new features (i.e., Generation). Finally, we acquire input features of the next layer through an activation function.

![Figure 2. An example of feature learning operation in FCN](image)

As noted, such a two-phase description of the feature learning process can be easily extended to CNN. Briefly, for the feature extraction of a CNN layer, a convolution operation is performed on each input feature with a kernel; then, for the feature generation, all the convolution output results are summed up to generate new input features for the next layer. For instance, there are \( p \) channels of inputs and \( m \) channels of outputs for layer \( i \). Hence, we employ \( p \times m \) kernels to extract the features from inputs \( x^{(i)} \). Using the operations of \( f_1 \) and \( f_2 \) described above, we can obtain the outputs \( x^{(i+1)} \) as new input features for layer \( i + 1 \).
3. RNN function-preserving transformation

In this section, the RNN feature learning process is reformulated according to the two-phase description provided in Section 2.3. Subsequently, with this reformulation, the RNN function-preserving transformations can be developed. These two transformations have been named as wider transformation and deeper transformation in this paper.

3.1. RNN reformulation

Different from spatial networks such as CNN and FCN, RNN is able to pass information through time. Besides receiving the information from input features at the current time period, the new RNN outputs also incorporate the output features from the last time period, as given in Eq. (1). Following the settings of the feature learning process in spatial networks, we assume that at time $t$, an RNN layer $i$ has $p$ input features $x_t^{(i)} = (x_{1,t}^{(i)}, x_{2,t}^{(i)}, ..., x_{p,t}^{(i)})$ and $m$ output features $x_t^{(i+1)} = (x_{1,t}^{(i+1)}, x_{2,t}^{(i+1)}, ..., x_{m,t}^{(i+1)})$. As shown, we have used a semicolon in subscript notation to distinguish the time information from other information. Subsequently, the input weight matrix is $W^{(i)} \in \mathbb{R}^{p \times m}$ and the output hidden state weight matrix is $H^{(i)} \in \mathbb{R}^{m \times m}$. For RNN, the meta operation $f_1$ is also a scalar multiplication and can be used to extract the information from $x_t^{(i)}$ and $x_t^{(i+1)}$. Specifically, we specify the extracted features of layer $i$ at time $t$ related to $W^{(i)}$ and $H^{(i)}$ as follows:

$$oW^{(i)}_{j,k,t} = f_1(x_{j,t}^{(i)}, W_{j,k}^{(i)}), \quad j = 1, 2, ..., p, k = 1, ..., m,$$

$$oH^{(i)}_{k,t} = f_1(x_{t}^{(i+1)}, H_{k,t}^{(i)}), \quad l = 1, 2, ..., m, k = 1, ..., m.$$  

After that, we obtain the feature extraction results $o^{(i)}_{k,t}$ of layer $i$ at time $t$:

$$o^{(i)}_{k,t} = (oW^{(i)}_{1,k,t}, ..., oW^{(i)}_{p,k,t}, oH^{(i)}_{1,k,t}, ..., oH^{(i)}_{m,k,t}), \quad k = 1, ..., m.$$  

By summing up the information from the extracted features with the use of aggregating function

$$f_2(o^{(i)}_{k,t}) = \sum_{j=1}^{p} oW^{(i)}_{j,k,t} + \sum_{i=1}^{m} oH^{(i)}_{l,k,t}, \quad \text{for} \; k = 1, ..., m,$$

we generate the new outputs $x_t^{(i+1)}$ at time $t$ through the following equation:

$$x_t^{(i+1)} = f_2\left(o^{(i)}_{i,t}ight), f_2\left(o^{(i)}_{i+1,t}ight), \ldots, f_2\left(o^{(i)}_{m,t}\right)).$$

An example of two-phase feature learning in RNN is depicted in Fig. 3. As shown, we have two input features at time $t$ and three output features at time $t-1$. Both input and output features perform the element-wise meta operation $f_1$ with its corresponding weight matrices, and then we acquire the extracted features. Subsequently, we utilize the aggregating function $f_2$ to generate new output features.
of RNN at time $t$. As noted, from the perspective of this two-phase feature learning process, the network transformation can be extended to not only RNN but also to other DL structures in temporal domains.

![Figure 3. An example of a two-phase feature learning operation in RNN](image)

According to Chen et al. (2016), a new set of network parameters $\theta'$ for a student neural network $G(x; \theta')$ (i.e., the extended neural network) with inputs, $x$ is identified to specify the same function as the teacher neural network $F(x; \theta)$ (i.e., the original neural network) with a set of network parameters $\theta$, and the function-preserving transformation is formulated by the following equation:

$$\forall x, F(x; \theta) = G(x; \theta').$$

Eq. (8) is an essential criterion 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 allows the student network to be trained based on the parameters of its teacher network as a warm start, which can save computational budgets. With this concept of function-preserving transformation, we present the RNN wider transformation and deeper transformation as follows.
3.2. Wider transformation

The wider transformation of a neural network influences at least two layers, which complicates the wider transformation. Suppose that an RNN layer \(i\) has \(p\) input and \(m\) output features, and RNN layer \(i + 1\) has \(q\) output features, then the parameters of layers \(i\) and \(i + 1\) are \(W^{(i)} \in \mathbb{R}^{p \times m}, H^{(i)} \in \mathbb{R}^{m \times m}\) and \(W^{(i+1)} \in \mathbb{R}^{m \times q}, H^{(i+1)} \in \mathbb{R}^{q \times q}\), respectively. If we widen layer \(i\) to layer \(i'\) with \(n\) output features with \(n > m\), then the original network parameters of layers \(i\) and \(i + 1\) are replaced by \(W^{(i') \in \mathbb{R}^{p \times n}}, H^{(i') \in \mathbb{R}^{n \times n}}\) and \(W^{(i' + 1)} \in \mathbb{R}^{n \times q}, H^{(i' + 1)} \in \mathbb{R}^{q \times q}\), separately.

To derive the wider transformation for RNN operations, we follow Chen et al. (2016) to specify a random mapping function \(g: \{1,2, ..., n\} \rightarrow \{1,2, ..., m\}\) that satisfies:

\[
g(k) = \begin{cases} 
  k & k \leq m \\
  \text{random sample from } \{1,2, ..., m\} & k > m
\end{cases}
\]  

(9)

Eq. (9) indicates how the features in a new student network layer have been represented. The first \(m\) features are exactly the same as those in its teacher network layer, while from the \((m + 1)\)th feature, they have been randomly replicated from the original features in the teacher network layer. To keep the function-preserving transformation, we utilize a replication factor, \(f_w\), similar to that in Chen et al. (2016):

\[
f_w(k) = 1 / |\{l | g(k) = g(l)\}|, l = 1,2, ..., n, k = 1,2, ..., n .
\]  

(10)

From Eq. (10), if a feature is replicated once, there will be two exactly identical features in the input 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 as follows:

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

\[
W^{(i')}_{j,k} = W^{(i)}_{j,g(k)}, \quad j = 1,2, ..., p, k = 1, ..., n,
\]

\[
H^{(i')}_{l,k} = f_w(l)H^{(i)}_{g(l),g(k)}, \quad l = 1,2, ..., n, k = 1, ..., n,
\]

\[
W^{(i' + 1)}_{k,h} = f_w(k)W^{(i + 1)}_{g(k),h}, \quad k = 1,2, ..., n, h = 1, ..., q,
\]

\[
H^{(i' + 1)}_{r,h} = H^{(i + 1)}_{r,h}, \quad r = 1,2, ..., q, h = 1, ..., q.
\]

Then, the function-preserving transformation for layer \(i\), i.e., Eq. (8), is satisfied.
**Proof:** First, we consider the feature extraction for the wider layer $i'$. We apply the new student network parameters of $W^{(i')}$ and $H^{(i')}$ in Proposition 1 to Eqs. (4) and (5), and the extracted features at time $t$ are given by:

$$oW_{j,k;t}^{(i')} = oW_{j,g(k);t}, \quad j = 1,2,\ldots,p, k = 1,\ldots,n,$$  \hspace{1cm} (11)

$$oH_{l;k;t}^{(i')} = f_w(l) oH_{g(l),g(k);t}, \quad l = 1,2,\ldots,n, k = 1,\ldots,n.$$  \hspace{1cm} (12)

Following Eq. (6), the feature extraction results of layer $i'$ at time $t$, $o_k^{(i')}$, are given by:

$$o_k^{(i')} = (oW_{1,k;t}^{(i')}, \ldots, oW_{p,k;t}^{(i')}, oH_{1,k;t}^{(i')}, \ldots, oH_{n,k;t}^{(i')})$, \quad k = 1,\ldots,n.$$

Then, by Eq. (7) and the aggregating function $f_2(o_k^{(i')}) = \sum_{l=1}^{p} oW_{j,g(k);t}^{(i')} + \sum_{l=1}^{n} oH_{l;k;t}^{(i')}$, $k = 1,\ldots,n$, the new output $x_{k;t}^{(i'+1)}$ are given by:

$$x_{k;t}^{(i'+1)} = f_a\left(f_2(o_k^{(i')})\right)$$

$$= f_a\left(\sum_{j=1}^{p} oW_{j,g(k);t}^{(i')} + \sum_{l=1}^{n} oH_{l;k;t}^{(i')}\right)$$

$$= f_a\left(\sum_{j=1}^{p} oW_{j,g(k);t}^{(i')} + \sum_{l=1}^{n} f_w(l) oH_{g(l),g(k);t}^{(i')}\right)$$

$$= f_a\left(\sum_{j=1}^{p} oW_{j,g(k);t}^{(i')} + \sum_{l=1}^{m} oH_{l,k;g(k);t}^{(i')}\right)$$

$$= x_{g(k);t}^{(i+1)}, \quad k = 1,\ldots,n,$$

where the third equality holds by Eqs. (11) and (12), the fourth equality holds by Eq. (10), and the last equality holds by the definition of the aggregating function and Eq. (7). Hence, the output features vector $x_t^{(i'+1)}$ of layer $i'$ also serve as the input features of layer $i' + 1$ following Eq. (7) can be identified as

$$x_t^{(i'+1)} = (x_{g(1);t}^{(i'+1)}, x_{g(2);t}^{(i'+1)}, \ldots, x_{g(n);t}^{(i'+1)}).$$  \hspace{1cm} (15)

Note that $x_t^{(i'+1)}$ is essentially a random mapping defined by $g$. Similar to the RNN reformulation in wider layer $i'$ as shown above, we leverage the new student network parameters of $W^{(i'+1)}$ and $H^{(i'+1)}$ in Proposition 1 to Eqs. (4) and (5), and acquire the extracted features for the next layer $i'$ + 1 as follows:

$$oW_{k,h;t}^{(i'+1)} = f_w(k) oW_{g(k),h;t}^{(i+1)}, \quad k = 1,2,\ldots,n, h = 1,\ldots,q,$$  \hspace{1cm} (16)

$$oH_{r,h;t}^{(i'+1)} = oH_{r,h;t}^{(i+1)}, \quad r = 1,2,\ldots,q, h = 1,\ldots,q.$$  \hspace{1cm} (17)
As observed in Eq. (17), the feature extraction for outputs of layer $i' + 1$ is the same as that of layer $i + 1$ since the dimension is unchanged. Following Eq. (6), the feature extraction results of layer $i' + 1$ at time $t$, $o_{h,t}^{(i' + 1)}$, are given by

$$o_{h,t}^{(i' + 1)} = \left( oW_{1,h,t}^{(i' + 1)}, ... oW_{n,h,t}^{(i' + 1)}, oH_{1,h,t}^{(i' + 1)}, ... oH_{q,h,t}^{(i' + 1)} \right), \quad h = 1, ..., q. \quad (18)$$

Next, we sum up the extracted features with the aggregating function $f_2\left(o_{h,t}^{(i' + 1)}\right) = \sum_{k=1}^{n} oW_{k,h,t}^{(i' + 1)} + \sum_{r=1}^{q} oH_{r,h,t}^{(i' + 1)}$, for $h = 1, ..., q$, to generate the new output $x_{h,t}^{(i' + 2)}$, $h = 1, ..., q$. Similar to the analysis of Eq. (14), by incorporating Eqs. (9-10) and Eqs. (16-18), we have the following output:

$$x_{h,t}^{(i' + 2)} = f_a\left(f_2\left(o_{h,t}^{(i' + 1)}\right)\right)$$

$$= f_a\left(\sum_{k=1}^{n} f_w(k) oW_{g(k),h,t}^{(i' + 1)} + \sum_{r=1}^{q} oH_{r,h,t}^{(i' + 1)}\right)$$

$$= f_a\left(\sum_{k=1}^{m} oW_{k,h,t}^{(i' + 1)} + \sum_{r=1}^{q} oH_{r,h,t}^{(i' + 1)}\right)$$

$$= x_{h,t}^{(i' + 2)}, \quad h = 1, ..., q. \quad (19)$$

Therefore, we have output features of layer $i' + 1$

$$x_{i',t}^{(i' + 2)} = \left( x_{1,t}^{(i' + 2)}, x_{2,t}^{(i' + 2)}, ..., x_{q,t}^{(i' + 2)} \right). \quad (20)$$

From Eq. (20), the output features vector $x_{i',t}^{(i' + 2)}$ of layer $i' + 1$ is identical to that of layer $i + 1$, i.e., $x_{i',t}^{(i' + 2)} = x_{i,t}^{(i' + 2)}$, in line with the function-preserving transformation in Eq. (8), which indicates that the wider transformation for layer $i$ is a function-preserving transformation. Hence, the proof of Proposition 1 is completed.

### 3.3. Deeper transformation

RNN deeper transformation is developed in a much more straightforward manner than RNN wider transformation. Similar to Chen et al. (2016), the deeper transformation should replace an RNN layer with two RNN layers utilizing the identity-mapping method. In other words, the input features of the new student RNN layer should be identical to that of the teacher RNN layer. Suppose we deepen an RNN layer $i$ with a new student layer $i''$. If layer $i$ has $m$ input features, then layer $i''$ has $m$ input and $m$ output features. The new student network parameters of layer $i''$ are $W^{(i'')} \in \mathbb{R}^{m \times m}$ and
\(H^{(i')}(t) \in \mathbb{R}^{m \times m}\). According to Chen et al. (2016), the deeper transformation is applicable when the activation function has this property, e.g., \(f_a(I \cdot f_a(z)) = f_a(z)\), where \(I\) is an identity matrix and \(z\) is a vector. 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 2. 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 2.** When deepening layer \(i\) with an RNN layer \(i'\) in a network, we can set the RNN layer parameters \(W_{i,k}^{(i')}\) and \(H_{i,k}^{(i')}\) as:

\[
W_{i,k}^{(i')} = \begin{cases} 
1 & l = k \\
0 & l \neq k
\end{cases}, \quad l = 1,2,...,n, k = 1,...,n,
\]

\[
H_{i,k}^{(i')} = 0, \quad l = 1,2,...,n, k = 1,...,n.
\]

Then, the function-preserving transformation for layer \(i\), i.e., Eq. (8), is satisfied.

**Proof:** First, we consider the feature extraction for the deeper layer \(i'\). We apply the new network parameters of \(W^{(i')}\) and \(H^{(i')}\) in Proposition 2 to Eqs. (4) and (5), and we get the extracted features for inputs at time \(t\) and outputs at time \(t + 1\) as follows:

\[
oW_{i,k,t}^{(i')} = f_1(x_{t,i}^{(i')}, W_{i,k}^{(i')}) = \begin{cases} 
x_{t,i}^{(i')} & l = k \\
0 & l \neq k
\end{cases}, \quad l = 1,2,...,n, k = 1,...,n, \tag{21}
\]

\[
oH_{i,k,t}^{(i')} = f_1(x_{t-1,i}^{(i'+1)}, H_{i,k}^{(i')}) = 0 \quad l = 1,2,...,n, k = 1,...,n. \tag{22}
\]

Based on Eq. (6), the feature extraction results of layer \(i'\) at time \(t\), \(o_{k,t}^{(i')}\), are described as

\[
o_{k,t}^{(i')} = (0,...,x_{k,t}^{(i')},...,0,0,...,0), \quad k = 1,...,n. \tag{23}
\]

By evaluating the aggregating function \(f_2(o_{k,t}^{(i')}) = \sum_{l=1}^n oW_{l,k,t}^{(i')} + \sum_{l=1}^n oH_{l,k,t}^{(i')} = x_{k,t}^{(i')},\) for \(k = 1,...,n\), the new output \(x_{k,t}^{(i'+1)}\) is then obtained:

\[
x_{k,t}^{(i'+1)} = f_a(f_2(o_{k,t}^{(i')})) = f_a(x_{k,t}^{(i')}), \quad k = 1,...,n. \tag{24}
\]

Therefore, we have:

\[
x_{t}^{(i'+1)} = x_{t}^{(i')}. \tag{25}
\]

As observed in Eq. (25), the input and output features are identical, verifying the function-preserving transformation for layer \(i\) in Eq. (8). Hence, the proof of Proposition 2 is completed. ■
4. IAAS framework development

In this section, we propose an IAAS framework to search the neural architecture during the training process. Primarily, this framework has two main components, network transformation operation and network transformation control. Network transformation operation is embedded into the network transformation control. In the following subsections, we first describe the overview of the IAAS framework and then introduce three network actors and the net pool utilized in the framework.

4.1 Overview of IAAS framework

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 and efficient implementation of network transformation.

With the sequence-to-sequence technique (Bahdanau et al. 2015), each string of neural architecture layer information 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 bi-directional LSTM network, is subsequently used to learn the representations of this matrix. Note that 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 Fig. 4. 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, which is then given to a wider or deeper actor for the next operation. For instance, if the selector actor determines to widen the neural architecture, the wider actor will then start to appropriately widen the architecture. 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 entered into a net pool with a limited size, which has been designed to reduce the possibility of searching a local optimal architecture. In this
manner, the IAAS framework takes a random neural architecture from the net pool for the next-round network transformation.

4.2 Actor networks

In this research study, we follow Cai et al. (2018) to utilize the RL-based meta controller method to implement network transformation operation. Specifically, we model the automatic control procedures of these three actors as sequential decision-making processes, where the state is the current network architecture. The action spaces of the three actors are the corresponding network transformation operations but have some differences, as presented below. The model forecasting performance on the training dataset is used to evaluate the reward signal, which is further exploited to update the meta-controller using REINFORCE algorithm (Williams 1992). The best-searched architecture is output with the maximized cumulative rewards. It can be noticed that the reward function should encourage the actions to generate a new network with improved performance. The traditional reward function in NAS is designed based on the classification accuracy, which is bounded within [0, 1] region. However, electricity forecasting is a regression task, and the forecasting accuracy is usually evaluated using metrics like rooted mean squared errors (RMSE) or mean average errors (MAE) with a region bound
of $[0, +\infty]$. Compared to MAE, RMSE gives a higher weight to large errors, which means RMSE is probably more useful when large approximate errors are occurring in the classification or regression tasks (Li and Zhao 2006). Due to the uncertain and intermittent nature of renewable energy and load demands in power systems, it is hard to build a very accurate electricity forecasting models. Therefore, we introduce a new reward function with RMSE, i.e., $\frac{1}{RMSE}$, by awarding 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. For instance, reducing RMSE value from 10 to 9 is more significant than reducing RMSE value from 100 to 99.

4.2.1. Selector actor

We introduce a selector actor that realizes the automated architecture search within our proposed IAAS framework. Fig. 5 presents the net structure of the selector actor, which involves a two-layer FCN with a sigmoid activation function as a classification operator. The probabilities of selecting decisions of "wider", "deeper", and "unchanged" transformations are the outputs of this two-layer FCN, which are denoted by $P(\text{wider})$, $P(\text{deeper})$, and $P(\text{unchanged})$, respectively, in Fig. 5. Instead of outputting the decision with the highest probability, we leverage a distribution sampling method. Specifically, we normalize these three probabilities to a discrete probability distribution and then sample a decision from this distribution as the final decision of the selector actor. Such a distribution sampling method ensures that the selector actor has a much more exploration ability in the actor learning procedure within the RL scheme. Without using this methodology, the selector actor decision can easily fall into a fixed decision output when several initial decisions are the same. Considering this scenario, we also add this distribution sampling method into the design of the wider actor and the deeper actor as discussed in the following subsections.
4.2.2 Wider actor

This actor has been designed to widen a network layer with more units. Fig. 6 presents the structure of the wider actor net. As shown in the figure, similar to the selector actor, a two-layer FCN with a sigmoid activation function is adopted to learn the features from the encoder network. The output of this sigmoid classifier is the probability of widening the layers. The classifier will output four probabilities for a four-layer neural architecture as depicted in Fig. 6. Instead of directly widening the layer with the highest probability; we normalize these probabilities and incorporate the normalized probability as a discrete probability distribution as presented in Fig. 5. With such a distribution, we can sample a decision to evaluate which layer should be widened eventually. However, it should be noted that such a widening procedure is suitable to that neural architecture, in which all the layers have the same type of structure. If the network is established with diversified types of structures, we should incorporate an additional step to reset the widening probability of each layer before using the distribution sampling method. For instance, if layer $i$ is an RNN layer and layer $i + 1$ is also an RNN layer, the widening probability for layer $i$ remains unchanged; if layer $i$ is an RNN layer and layer $i + 1$ is a CNN layer, we then set the widening probability of layer $i$ as 0. Subsequently, we check layer $i + 1$ and layer $i + 2$ to adjust the widening probability of layer $i + 1$ till the last network layer.
After evaluating which layer to be widened, the wider actor widens the determined layer based on its unit amount. Specifically, we set a widening rule with a self-defined sequence \(e.g., [4, 8, 16, 32, \ldots, K, K + 16]\). 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. Note that 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 more accommodating than Chen et al. (2021b) since we do not need to manually set the maximum widening units. Even though adding more units in a layer has a chance to improve the network performance, it also would cause the over-fitting issue. With the application of the RL technique, the decision-making actor is likely to evaluate whether the network must be further widened by observing the cumulative rewards.

Figure 6. Illustration of the wider actor net with an input neural architecture with four layers. (Note: all the four layers have the same type of neural structure in this example; P indicates the probability).

4.2.3 Deeper actor

This section proposes a powerful deeper actor to deepen the given network. Specifically, we have three-layer candidates, namely, FCN layer, CNN layer, and RNN layer. In the process of actor training, we
believe any layer of the given network architecture can be deepened. For comparison, the Net2Deeper actor in the EAS framework set a limitation that an FCN layer can only be added into a given network on top of all CNN and pooling layers. Therefore, to make the deepening process more flexible, the deeper actor is designed with the structure presented in Fig. 7. As shown, the learned features from the encoder network are provided to an RNN structure with a sigmoid activation function. There are two implementation steps of the deeper actor. The first step is to determine which layer type (e.g., FCN, CNN, or RNN layer) is utilized for the network deepening. Subsequently, the decision of type is made using the distribution sampling method introduced above. The second step is to evaluate the deepening place using the decision of type and learned features from the encoder network. Briefly, we index each layer of the given network and calculate the probability for inserting the determined layer. Then, the given network can be deepened with the distribution sampling method. As we can observe, we set the kernel size in CNN to three since 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. 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. Moreover, for an inserted student RNN 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 $168 \times 3$, 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 3.
4.3 Net pool

A net pool is derived in our proposed IAAS framework to construct a space for the network structure exploration. As presented in Fig. 4, the updated network enters into a net pool after the network transformation operation, which contains several network architectures. For the next-round transformation, the designed net pool will randomly output one network architecture. To search the network architecture space efficiently, we set the capacity of our network pool as $C_0$, and utilize a general algorithm to manage networks in the pool, which has been presented below. This algorithm essentially ranks the network architectures based on their forecasting performance. For example, if there are six neural architectures in the net pool and the value of $C_0$ is set as five, a neural architecture with the lowest performance will be kicked out from the net pool. Notably, the initial net pool is randomly constructed from networks such as FCN, CNN, and RNN. With such a dynamic algorithm, the pool will efficiently have a set of networks with good performances after network transformations of several rounds. Moreover, it is also possible to avoid the local optimal state of the searched architecture. 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 managed. This stopping criterion is problem-dependent. In this work, this criterion is set to a fixed number of searches (e.g., 200) to validate the performance of our search scheme.
Algorithm 1. Net pool management

**Requires:** Network pool set, \( Q \); Net pool capacity, \( C_Q \); number of networks in net pool, \( S_Q \); maximum number of search episodes, \( N \); test dataset \( (X, Y) \), \( X \) is the input data, \( Y \) is the output data.

**Algorithm:**
1. Initialize network pool set \( Q \) from a network layer;
2. Train and test networks in \( Q \);
3. \( CNT = 0 \)
4. while \( \text{STOP (CNT)} \) :  
   \( \triangleright \) loop until the stopping criterion is satisfied  
   4.1 \( CNT = CNT + 1 \)
   4.2 Train networks in \( Q \)
   4.3 for \( i = 1 \) to \( \text{size}(Q) \):
   4.4 \( q = Q_i \)
   4.5 \( M_i = \text{PERFORMANCE (q)} \)
   4.6 Sort networks in \( Q \) by their performance \( M_i(i \in \{1,2,...,\text{size}(Q)\}) \);
   4.7 if \( S_Q > C_Q \):
      Drop the last \( S_Q - C_Q \) networks;
5. return \( Q \)

6. function \( \text{STOP (CNT)} \)  
   \( \triangleright \) stopping criterion in our setting
7. return \( CNT < N \)

8. function \( \text{PERFORMANCE (q)} \)  
   \( \triangleright \) performance metric in our setting
9. \( \hat{Y} = q(X) \)
10. return \( \text{RMSE (\hat{Y}, Y)} \)

5. Numerical Results

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

5.1 Data description

We performed 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 datasets of Maine (ME) and New Hampshire (NH) for the year 2020 are collected from ISO New England Inc\textsuperscript{1}. Similarly, for the wind power forecasting experiment, we utilize two wind farm (WF) datasets from a Chinese wind power company. Due to the fact that the energy-based time-series data are typically influenced by various seasons (Qin et al. 2021 and Jalali et al. 2021a), we classify each dataset depending on the season, \textit{i.e.}, spring, summer, autumn, and winter, respectively. Consequently, there are sixteen independent subsets for our experimental evaluations.

In each electricity load subset, the time step of load data is set to an hour, thereby leading to 24 data points per day, and thus approximately 2160 data points per season. Following Jalali \textit{et al.} (2021a), we split each independent subset into a training dataset with 75\% data points and a test dataset with 25\% data points. Moreover, the data in the training dataset is chronically earlier than that in the test dataset. When forecasting the data points in the load demand subsets, apart from using historical load demand data, we incorporate their corresponding time information as inputs, namely $h^{th}$ hour of a day and $w^{th}$ day of a week, where $h$ and $w$ are restricted to ranges of $[1, 24]$ and $[1, 7]$, respectively. For instance, if the load demand is 50 MW at 8:00 am on a Sunday, we describe such an input vector as $(50, 8, 7)'$. To improve forecasting accuracy, we use one-week data with the size of $168 \times 3$ as one input for the forecasting model to predict the 24h-ahead load demand point.

In each wind power subset, the time step of wind power data is set as an hour. Similar to the settings in Zhang \textit{et al.} (2020), the last five-day data of each season are treated as the test dataset, and the remaining data of that season are the training dataset. As for the forecast of data points in the wind power subsets, numerical weather prediction (NWP) data are widely utilized following the day-ahead wind power literature (see Zhang \textit{et al.} 2020 and Chen \textit{et al.} 2021a). We can notice that the NWP data contains five dimensions, namely, humidity, wind speed, wind direction, temperature and air pressure, which are weather simulation data generated from NWP computer models of the atmosphere and oceans (Rabier 2005). Consistent with Zhang \textit{et al.} (2020), we use the sine and cosine values of wind direction to signify the real wind direction. Moreover, we use the historical wind speed, wind power data, and hourly timestamp information in a day as additional input variables. Therefore, we have an input vector
with nine dimensions. We use three-day data with a size of $72 \times 9$ as one input for the forecasting model to predict the 24h-ahead wind power point. Unlike the load input data, NWP data at the forecasting point is known in advance. For instance, if we are at 8 am on Jun 9 and forecast the wind power generation at 8 am on Jun 10, we can use the NWP data from 9 am on Jun 8 to 8 am on Jun 10 as input.

Figure 8. Data description of four cases: (a) ME load dataset, (b) NH load dataset, (c) WF1 wind power dataset, and (d) WF2 wind power dataset.

We plot the dataset by seasons in each sub-figure of Fig. 8. Moreover, we plot the data density distribution plot and data boxplot in each sub-figure of Fig. 8. We can observe that the load data is quite different from wind power data; however, both the load data and wind power data have seasonal patterns. For instance, the ME-spring load data has similar patterns to the NH-spring load data, and WF1-winter wind power generation data is also akin to WF2-winter wind power generation data. However, despite these similarities, they also have some differences from the further observations. We select these datasets for the numerical experiments for the following reasons. First, two types of electricity energy data, \textit{i.e.}, load and wind power data, are utilized to demonstrate the generalizations of the proposed IAAS framework. Second, the two identical types of data within some similarities are leveraged to display the stability of this framework.
5.2 Experimental settings and performance metrics

Comparison experiments are imperative for the demonstrations of forecasting model performance. In this research study, we utilize ten baselines including three classical ML methods, i.e., support vector regression (SVR), random forest (RF), and ridge regression (RR). Moreover, we use the following seven DL models to showcase the advantages of our proposed IAAS framework. Firstly, we utilize three traditional DL methods, which involve 3-layer CNN, 3-layer LSTM network, and 2-layer CNN plus 1-layer LSTM network. Secondly, we exploit three proposed networks in the literature, where ResNet and ResNetPlus were derived from Chen et al. (2019) and deep adaptive input normalization (DAIN) was introduced by Passalis et al. (2020). Briefly, Chen et al. (2019) proposed two residual neural networks, where ResNetPlus was developed based on ResNet by utilizing a side block technique; Passalis et al. (2020) introduced the DAIN method for the time-series data in order to build a better forecasting model. Thirdly, we employ the latest SNAS framework (Chen et al. 2021b), which targets the MTF model development with NAS technique.

The experimental settings including all parameters and values for the baseline models (or methods) and our proposed IAAS framework are listed in Table 1. As introduced above, we use the ReLU activation function to fulfill the identity mapping requirement of the deeper transformation in the IAAS framework and employ it to all the DL baseline models. We set the batch size for all the networks to 256 by considering both the computation and model generalization issues (He et al. 2019). Moreover, we also use Adam optimizer in DL-based baselines and our IAAS framework since this optimizer can adaptively adjust the learning rate and the momentum of parameters (Kingma and Ba 2017). For the three traditional machine learning methods, we employ the default optimization algorithms in the Python package, "scikit-learn", to optimize their parameters.

Table 1. Experimental settings for the baselines and IAAS framework

| Model | Parameters          | Value |
|-------|---------------------|-------|
| CNN   | Activation function | ReLU  |
|       | Batch size          | 256   |
|       | Optimizer           | Adam  |
|       | No. of kernels      | 32    |
|       | No. of layers       | 3     |
| LSTM  | Activation function | ReLU  |
|       | Batch size          | 256   |
|       | Optimizer           | Adam  |
|       | Hidden size         | 24    |
| Model         | No. of layers | Activation function | Batch size | Optimizer | Hidden size | No. of kernels | No. of CNN layers | No. of LSTM layer |
|--------------|---------------|---------------------|------------|-----------|-------------|---------------|------------------|------------------|
| CNN-LSTM     | 3             | ReLU                | 256        | Adam      | 8           | 16            | 2                | 1                |
| ResNet       | 3             | ReLU                | 256        | Adam      | 3           | 16            |                  |                  |
| ResNetPlus   | 3             | ReLU                | 256        | Adam      | 3           | 16            |                  |                  |
| DAIN         | 3             | ReLU                | 256        | Adam      | 1e-5        | 1e-3          | 1e-4             |                  |
| SNAS         | 3             | ReLU                | 256        | Adam      | 32          | 2             | 64               | 500              |
| IAAS         | 3             | ReLU                | 256        | Adam      | 50          | 500           |                  |                  |

Based on the existing literature in both load forecasting and wind power forecasting (e.g., Cevik et al. 2019, Chen et al. 2019, Chen et al. 2021a), we use two evaluation metrics, namely RMSE and MAE, to quantify the performance of forecasting results:

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

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |Y_i - \hat{Y}_i|,
\]

where \(Y\) is the actual value and \(\hat{Y}\) is the predicted value, and \(N\) is the number of forecasting time points.
5.3 Results and analysis

5.3.1. 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 dataset is split by four seasons, we thus obtain eight network structures of load forecasting models as presented in Table 2. To describe the architecture of each model, we leverage an example of the ME-spring case for illustration. ME-spring model is a six-layer network model: both the first and second layers are RNNs with four output units; the third layer is a CNN layer with four output channels; the fourth to sixth layers are FCN layers with sixteen, eight, and four output units, respectively. As observed, all these eight network structures are different from each other, with the largest one of twenty-three layers (see NH-winter case) and the smallest one of three layers (see ME-autumn case). Since 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 dataset by generating an appropriate structure. Moreover, we find that all the eight network models are using RNN layers, denoting the significance of incorporating RNN in our proposed IAAS framework to obtain the high-quality network structure.

Table 2. Network structures of load demand forecasting experiments using IAAS framework

| Cases      | Network structures                                      |
|------------|---------------------------------------------------------|
| ME-spring  | rnn-4 $\rightarrow$ rnn-4 $\rightarrow$ conv-4 $\rightarrow$ fc-16 $\rightarrow$ fc-8 $\rightarrow$ fc-4 |
| ME-summer  | fc-12 $\rightarrow$ fc-4 $\rightarrow$ rnn-3 $\rightarrow$ rnn-3 |
| ME-autumn  | rnn-16 $\rightarrow$ rnn-4 $\rightarrow$ fc-4 |
| ME-winter  | conv-3 $\rightarrow$ conv-3 $\rightarrow$ fc-16 $\rightarrow$ fc-16 $\rightarrow$ fc-20 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ fc-16 $\rightarrow$ rnn-3 $\rightarrow$ fc-4 $\rightarrow$ rnn-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ rnn-3 |
| NH-spring  | conv-3 $\rightarrow$ rnn-3 $\rightarrow$ rnn-4 $\rightarrow$ conv-4 $\rightarrow$ conv-4 $\rightarrow$ fc-168 $\rightarrow$ fc-4 |
| NH-summer  | conv-3 $\rightarrow$ rnn-16 $\rightarrow$ fc-16 |
| NH-autumn  | conv-4 $\rightarrow$ rnn-6 $\rightarrow$ fc-12 $\rightarrow$ fc-16 |
| NH-winter  | fc-4 $\rightarrow$ fc-4 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-4 $\rightarrow$ conv-3 $\rightarrow$ fc-4 $\rightarrow$ conv-3 $\rightarrow$ fc-4 $\rightarrow$ conv-3 $\rightarrow$ rnn-3 $\rightarrow$ conv-4 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-4 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-3 $\rightarrow$ conv-8 $\rightarrow$ conv-4 $\rightarrow$ conv-8 $\rightarrow$ conv-3 $\rightarrow$ fc-4 |

Fig. 9 depicts the network transformation procedures of ME-summer and NH-spring cases using the IAAS framework. As shown in Fig. 9(a), the final network of the ME-summer case is achieved after four transformations. Specifically, the network is initially made up of two FCN layers with four output units in each layer. After training the first network, the selector actor determines to widen the network, and the wider actor then widens the first layer with another four units. Similarly, after training the
second network, the selector actor determines to deepen the network. The deeper actor then adds an RNN layer with three output units after the second FCN layer since the input dimension is three. Furthermore, after training the third network, the selector actor determines to widen the network, and the wider actor widens the first FCN layer with another four units. After training the fourth network, the selector actor determines to deepen the network, and the deeper actor deepens the network with another three-output-unit RNN layer. Finally, after training the fifth network, the selector actor determines to stop the transformation, and the optimized network architecture is then acquired as an output. In Fig. 9(b), the process of the network transformation in the NH-spring case is similar to that in Fig. 9(a); however, it should be noted that when the deeper actor makes a decision, it can deepen any network part as we introduced in Section 4. For example, in Fig. 9(b), after training the first network, the selector actor determines to deepen the network, and the deeper actor then adds a CNN layer with four output channels behind the RNN layer of the first network. Such an implementation demonstrates the flexibility of the deeper actor in our proposed IAAS framework.

The forecasting accuracy results of our proposed IAAS framework and ten baseline models or methods with two evaluation metrics are presented in Table 3. 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, IAAS_LoadNet models perform the best in all the cases; as for MAE, IAAS_LoadNet models exhibit the best performance in almost all the cases, except in the case of ME-winter and NH-winter. However, IAAS_LoadNet models have close forecasting accuracies...
compared to the best models in those two cases. Moreover, we present the average forecasting accuracy of eight cases at the bottom row of Table 3 and determine that the IAAS_LoadNet achieves much better performance than all the baseline models on average. In detail, the RMSE and MAE for the IAAS_loadNet model are 85.927 and 68.777, respectively, and they are higher than the second-best model (ResNetPlus) with 20.2% (RMSE) and 19.4% (MAE), individually. In addition, we plot the predicted and actual loads for the test datasets of both ME and NH cases in Fig. 10. As we can witness from the figure, our proposed IAAS framework can match the actual loads in 24h-ahead points very well.

Figure 10. Comparison results between predicted electricity load and actual electricity load in eight testing sets: the red solid line represents the actual load values, and the blue dashed line represents the predicted values.
To further explore the stability of forecasting performance in the load forecasting experiments, we use the boxplot to demonstrate the variations in the forecasting accuracies, which are shown in Fig. 11. As we can observe, all the eleven models lead to various stabilities in terms of the two metrics; however, our proposed IAAS_LoadNet achieves the competitively stable accuracies in terms of RMSE and MAE. Such results also denote the necessity of using our IAAS framework to design a network architecture for various datasets. Based on this result and the forecasting accuracies results, we conclude that the proposed IAAS framework is able to adapt to the training data automatically for the development of load forecasting models. Moreover, IAAS framework can achieve the best load forecasting accuracies compared to the baseline models.

Figure 11. Boxplot of load demand forecasting accuracies
| Case       | Matrices | CNN-LSTM | CNN   | LSTM  | SVR    | RF    | RR    | ResNet | ResNetPlus | DAIN | SNAS | IAAS_LoadNet |
|------------|----------|----------|-------|-------|--------|-------|-------|--------|------------|------|------|--------------|
| ME-spring  | RMSE     | 92.185   | 129.637 | 85.889 | 127.303 | 91.743 | 78.186 | 90.474 | 67.359     | 160.01 | 68.185 | 66.915       |
|            | MAE      | 77.179   | 106.389 | 67.488 | 108.892 | 71.922 | 65.924 | 75.981 | 54.081     | 141.047 | 53.141 | 53.581       |
| ME-summer  | RMSE     | 183.065  | 253.069 | 178.769 | 205.587 | 150.13 | 174.829 | 176.666 | 136.877    | 189.011 | 99.044 | 84.673       |
|            | MAE      | 157.168  | 219.831 | 144.834 | 174.503 | 110.185 | 156.274 | 154.426 | 117.263    | 159.949 | 78.945 | 67.12        |
| ME-autumn  | RMSE     | 61.861   | 105.654 | 113.305 | 76.297  | 78.712 | 86.912 | 61.658 | 61.78       | 109.791 | 63.561 | 42.588       |
|            | MAE      | 47.967   | 83.125  | 99.871  | 64.851  | 60.825 | 68.879 | 47.603 | 45.101      | 86.19   | 49.676 | 31.805       |
| ME-winter  | RMSE     | 116.444  | 139.681 | 126.799 | 158.224 | 122.665 | 122.667 | 178.63  | 101.191     | 174.818 | 187.361 | 98.275       |
|            | MAE      | 92.108   | 110.551 | 102.411 | 133.083 | 97.17  | 102.815 | 160.755 | 77.459      | 149.517 | 168.771 | 81.774       |
| NH-spring  | RMSE     | 96.616   | 134.143 | 87.762  | 135.299 | 103.991 | 83.675 | 120.844 | 81.219      | 126.042 | 104.411 | 74.098       |
|            | MAE      | 78.98    | 111.162 | 66.197  | 115.136 | 80.32  | 64.861 | 104.666 | 62.556      | 104.477 | 81.621 | 58.841       |
| NH-summer  | RMSE     | 344.107  | 263.06  | 185.163 | 309.113 | 245.98 | 174.918 | 202.867 | 217.784     | 233.822 | 291.346 | 163.183      |
|            | MAE      | 300.204  | 209.384 | 143.672 | 250.066 | 179.979 | 143.961 | 168.763 | 174.775     | 184.384 | 267.53 | 133.351      |
| NH-autumn  | RMSE     | 129.845  | 144.704 | 110.334 | 151.338 | 111.243 | 173.333 | 154.397 | 87.04       | 132.726 | 88.219 | 61.175       |
|            | MAE      | 104.435  | 117.91  | 88.781  | 130.768 | 82.479 | 142.86  | 120.496 | 70.348      | 104.664 | 72.953 | 48.896       |
| NH-winter  | RMSE     | 125.705  | 173.151 | 97.386  | 156.389 | 118.505 | 107.466 | 129.619 | 107.948     | 120.588 | 110.047 | 96.505       |
|            | MAE      | 101.369  | 139.787 | 78.788  | 130.638 | 92.231 | 85.716  | 109.834 | 87.482      | 95.597  | 94.049 | 80.885       |
| Average    | RMSE     | 143.729  | 167.887 | 123.176 | 164.944 | 127.871 | 125.248 | 139.394 | 107.650     | 155.851 | 126.522 | 85.927       |
|            | MAE      | 117.903  | 134.853 | 97.647  | 136.038 | 95.873 | 102.131 | 119.375 | 85.284      | 126.367 | 108.336 | 68.777       |
5.3.2. Wind power forecasting experiment

As described above, we use two WF datasets considering the seasonal effects of conducting the wind power forecasting experiments, which constitutes eight cases in total. Table 4 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, and all the eight cases utilize the RNN structure. We present the forecasting accuracies of wind power with the two evaluation metrics in Table 5. It can be observed that all the developed IAAS_WindNet models have shown the best performance in terms of RMSE and MAE. Moreover, the average RMSE and MAE for the IAAS_WindNet model are 4.38 and 3.261, respectively, while those for the second-best model (ResNetPlus model) are 5.811 and 4.477, respectively. As presented in the table, our IAAS_WindNet models show better performance than the second-best models with 24.6% (RMSE) and 27.2% (MAE), separately.

Table 4. Network structures of wind power forecasting experiments using IAAS framework

| Cases     | Network structure |
|-----------|--------------------|
| WF1-spring| conv-6 $\rightarrow$ rnn-6 $\rightarrow$ fc-6 $\rightarrow$ fc-12 |
| WF1-summer| conv-16 $\rightarrow$ rnn-16 $\rightarrow$ fc-16 $\rightarrow$ fc-12 |
| WF1-autumn| conv-4 $\rightarrow$ rnn-6 $\rightarrow$ fc-4 $\rightarrow$ fc-16 |
| WF1-winter| rnn-12 $\rightarrow$ rnn-6 $\rightarrow$ rnn-6 $\rightarrow$ fc-16 $\rightarrow$ fc-6 |
| WF2-spring| conv-16-3 $\rightarrow$ rnn-4 $\rightarrow$ fc-6 $\rightarrow$ fc-4 |
| WF2-summer| rnn-9 $\rightarrow$ conv-24 $\rightarrow$ conv-12 $\rightarrow$ fc-6 $\rightarrow$ fc-16 $\rightarrow$ fc-6 |
| WF2-autumn| rnn-9 $\rightarrow$ conv-16 $\rightarrow$ conv-4 $\rightarrow$ fc-4 $\rightarrow$ fc-16 $\rightarrow$ fc-12 |
| WF2-winter| rnn-9 $\rightarrow$ conv-9 $\rightarrow$ rnn-12 $\rightarrow$ rnn-6 $\rightarrow$ fc-168 $\rightarrow$ fc-6 $\rightarrow$ fc-4 |

Consistent with the load forecasting experiments, we compare the predicted and actual wind power for both WF datasets in Fig. 12 and find out IAAS_WindNet is likely to generate an accurate 24h-ahead forecasting wind power. However, it should be noted that due to more uncertain and intermittent nature of wind power compared to that of electricity load, sometimes, it is challenging to accurately forecast the 24h-ahead wind power generation as shown in Fig. 12. We also draw the boxplot to evaluate the forecasting results' stability in these two evaluation metrics in Fig. 13. As shown in the figure, the IAAS_WindNet also achieves a competitively stable performance, compared to the others. Hence, we conclude that the proposed IAAS framework can obtain better forecasting models for the 24h-ahead wind power forecasting scenario as compared to the existing models or methods.
Figure 12. Comparison results between predicted wind power and actual wind power in eight testing sets: the red solid line represents the actual wind power, and the dashed blue line represents the predicted wind power.

Figure 13. Boxplot of wind power forecasting accuracies
Table 5. Forecasting accuracy results of wind power forecasting experiments

| Case     | Matrices | CNN-LSTM | CNN  | LSTM  | SVR   | RF    | RR    | RES   | RESPLUS | DAIN | SNAS | IAAS-WindNet |
|----------|----------|----------|------|-------|-------|-------|-------|-------|----------|------|------|--------------|
| WF1-spring | RMSE     | 6.368    | 6.971| 7.992 | 8.937 | 6.146 | 5.836 | 6.224 | 6.12     | 7.423| 5.689| **5.046**    |
|          | MAE      | 4.603    | 5.144| 6.176 | 7.769 | 4.494 | 4.67  | 5.02  | 4.81     | 5.925| 4.178| **3.79**     |
| WF1-summer | RMSE     | 5.938    | 3.751| 5.822 | 4.551 | 4.361 | 4.498 | 4.122 | 4.009    | 3.941| 4.562| **3.311**    |
|          | MAE      | 3.902    | 2.79 | 5.323 | 3.826 | 3.074 | 3.513 | 3.565 | 3.391    | 2.851| 3.271| **2.422**    |
| WF1-autumn | RMSE     | 3.297    | 4.126| 8.238 | 8.579 | 4.522 | 4.228 | 3.8   | 3.249    | 3.645| 4.296| **2.77**     |
|          | MAE      | 2.504    | 3.18 | 6.343 | 5.601 | 3.397 | 3.305 | 2.595 | 2.512    | 2.664| 2.686| **1.83**     |
| WF1-winter | RMSE     | 18.013   | 7.359| 17.526| 19.577| 7.81  | 6.279 | 5.816 | 5.52     | 8.347| 8.07 | **4.715**    |
|          | MAE      | 16.783   | 5.95 | 16.274| 18.433| 6.117 | 5.186 | 3.961 | 4.408    | 6.698| 6.614| **3.618**    |
| WF2-spring | RMSE     | 13.667   | 6.493| 13.574| 13.725| 8.538 | 6.242 | 7.606 | 7.347    | 8.358| 8.092| **5.566**    |
|          | MAE      | 11.264   | 5.034| 10.942| 10.438| 6.989 | 4.933 | 5.698 | 5.282    | 6.166| 6.221| **4.306**    |
| WF2-summer | RMSE     | 5.498    | 5.63 | 5.248 | 6.309 | 6.46  | 8.892 | 5.466 | 5.813    | 5.716| 7.6  | **4.507**    |
|          | MAE      | 4.682    | 4.705| 4.299 | 4.462 | 4.994 | 6.997 | 4.643 | 4.422    | 4.254| 5.643| **3.323**    |
| WF2-autumn | RMSE     | 8.491    | 8.639| 12.722| 15.813| 7.914 | 8.541 | 7.574 | 8.028    | 11.257|15.736| **5.133**    |
|          | MAE      | 6.617    | 6.639| 9.773 | 11.592| 6.054 | 6.543 | 5.816 | 5.69     | 8.13 | 11.519| **4.043**    |
| WF2-winter | RMSE     | 24.042   | 9.518| 23.488| 28.347| 7.146 | 7.906 | 7.906 | 6.405    | 11.054|16.028| **3.989**    |
|          | MAE      | 23.222   | 7.646| 22.664| 27.584| 5.306 | 6.495 | 5.756 | 5.304    | 8.668| 15.123| **2.753**    |
| Average  | RMSE     | 10.664   | 6.561| 11.826| 13.23 | 6.612 | 6.553 | 6.064 | 5.811    | 7.468| 8.759| **4.38**     |
|          | MAE      | 9.197    | 5.136| 10.224| 11.213| 5.053 | 5.205 | 4.632 | 4.477    | 5.67 | 6.907| **3.261**    |
5.4. Ablation studies

As discussed in Section 4, the primary differences between the IAAS and the EAS frameworks are that the IAAS firstly utilizes a selector actor to intelligently determine how to transform the network instead of any manual process, and secondly it employs a net pool to obtain a set of networks with good performance as a buffer. To demonstrate the advantages of these two novel features, we conduct ablation experiments in this subsection 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. As we can observe that the IAAS_sn is similar to the EAS framework (Cai et al. 2018). However, the EAS framework is applied to the image data, which is quite different from the energy-based time-series data. Subsequently, we reset the manual parameters that the RL-based meta-controllers take three wider steps and three deeper steps at each search episode. Meanwhile, we leverage this manual setting for the IAAS_s framework. Keeping the other hyperparameter settings the same, we obtain the forecasting accuracy results as presented in Tables 6-7. These results indicate that, for the load demand forecasting experiments, IAAS_LoadNet has much better performance as compared to IAAS_n_LoadNet and IAAS_sn_LoadNet, in terms of the average RMSE and MAE values. Specifically, IAAS_LoadNet performs better than IAAS_n_LoadNet with 11.8% (RMSE) and 10.4% (MAE), whereas IAAS_LoadNet is also better than IAAS_sn_LoadNet with 12.8% (RMSE) and 10.8% (MAE). For the wind power forecasting experiments, IAAS_WindNet also provides much better performance than IAAS_n_WindNet and IAAS_sn_WindNet regarding forecasting accuracy. On average, IAAS_WindNet is better than IAAS_n_WindNet with 8.3% (RMSE) and 11.6% (MAE), and is better than IAAS_sn_WindNet with 27.8% (RMSE) and 28.5% (MAE). However, IAAS_LoadNet and IAAS_WindNet provide close performance compared with IAAS_s_LoadNet and IAAS_s_WindNet, respectively. To further differentiate the performance of these two pairs (IAAS_LoadNet & IAAS_s_LoadNet, IAAS_WindNet & IAAS_s_WindNet) in forecasting accuracy, we conduct a pair \( t \)-test, and the results have been presented in Table 8. As we can observe, the \( p \) value in each comparison
is much greater than 0.05, which indicates the IAAS_LoadNet and IAAS_s_LoadNet have the same performances, statistically, as well as IAAS_WindNet and IAAS_s_WindNet.

Table 6. Load forecasting accuracy results in ablation experiments

| Case       | Matrices | IAAS_LoadNet | IAAS_s_LoadNet | IAAS_n_LoadNet | IAAS_sn_LoadNet |
|------------|----------|--------------|----------------|----------------|-----------------|
| ME-spring  | RMSE     | 66.915       | 64.754         | 67.32          | 71.081          |
|            | MAE      | 53.581       | 51.466         | 52.908         | 57.266          |
| ME-summer  | RMSE     | 84.673       | 93.671         | 122.912        | 122.499         |
|            | MAE      | 67.12        | 73.577         | 95.754         | 100.382         |
| ME-autumn  | RMSE     | 42.588       | 44.446         | 46.715         | 44.906          |
|            | MAE      | 31.805       | 32.985         | 35.724         | 35.613          |
| ME-winter  | RMSE     | 928.275      | 89.317         | 106.324        | 101.344         |
|            | MAE      | 81.774       | 72.046         | 86.221         | 78.829          |
| NH-spring  | RMSE     | 74.098       | 72.031         | 104.491        | 77.867          |
|            | MAE      | 58.841       | 57.264         | 85.321         | 62.195          |
| NH-summer  | RMSE     | 163.183      | 163.074        | 165.328        | 192.403         |
|            | MAE      | 133.351      | 133.607        | 130.966        | 148.7           |
| NH-autumn  | RMSE     | 61.175       | 66.882         | 64.734         | 74.281          |
|            | MAE      | 48.896       | 51.332         | 52.513         | 58.552          |
| NH-winter  | RMSE     | 96.505       | 96.038         | 101.467        | 103.911         |
|            | MAE      | 80.885       | 77.456         | 81.41          | 81.811          |
| Average    | RMSE     | 85.926       | 86.277         | 97.411         | 98.536          |
|            | MAE      | 69.531       | 68.717         | 77.602         | 77.918          |

Table 7. Wind power forecasting accuracy results in ablation experiments

| Case       | Matrices | IAAS_LoadNet | IAAS_s_LoadNet | IAAS_n_LoadNet | IAAS_sn_LoadNet |
|------------|----------|--------------|----------------|----------------|-----------------|
| WF1-spring | RMSE     | 5.046        | 4.876          | 5.162          | 6.318           |
|            | MAE      | 3.79         | 3.799          | 4.021          | 4.397           |
| WF1-summer | RMSE     | 3.311        | 2.433          | 3.211          | 3.173           |
|            | MAE      | 2.422        | 1.751          | 2.477          | 2.269           |
| WF1-autumn | RMSE     | 2.77         | 2.796          | 2.86           | 4.358           |
|            | MAE      | 1.83         | 2.104          | 2.2            | 2.887           |
| WF1-winter | RMSE     | 4.715        | 4.742          | 5.123          | 8.772           |
|            | MAE      | 3.618        | 3.732          | 3.924          | 7.208           |
| WF2-spring | RMSE     | 5.566        | 5.461          | 5.982          | 6.375           |
|            | MAE      | 4.306        | 3.951          | 4.483          | 4.64            |
| WF2-summer | RMSE     | 4.507        | 4.628          | 4.968          | 5.344           |
|            | MAE      | 3.323        | 3.343          | 3.823          | 3.978           |
| WF2-autumn | RMSE     | 5.133        | 6.404          | 6.788          | 8.86            |
|            | MAE      | 4.043        | 4.762          | 5.358          | 6.917           |
| WF2-winter | RMSE     | 3.989        | 4.269          | 4.1            | 5.325           |
|            | MAE      | 2.753        | 3.051          | 3.22           | 4.206           |
| Average    | RMSE     | 4.38         | 4.451          | 4.774          | 6.065           |
|            | MAE      | 3.261        | 3.312          | 3.688          | 4.563           |
Table 8. Paired t-test results between IAAS model and IAAS_s model (\(\alpha = 0.05\))

|                  | IAAS_LoadNet v.s. IAAS_s_LoadNet | IAAS_WindNet v.s. IAAS_s_WindNet |
|------------------|-----------------------------------|----------------------------------|
| \(p_{RMSE}\)     | 0.86                              | 0.744                            |
| \(p_{MAE}\)      | 0.641                             | 0.743                            |

Note: \(p\) indicates the \(p\) value in statistics.

Moreover, we present the parameter amount of the obtained models in each scenario to indicate the model structural complexity in Tables 9-10. As shown in these results, our proposed IAAS_LoadNet model uses the least average amount of parameters among all four models. On average, IAAS_LoadNet only requires 5340.125 parameters but IAAS_s_LoadNet, IAAS_n_LoadNet and IAAS_sn_LoadNet require 272551.375, 23497.25, 96915.625 parameters, separately. In case of wind power, the IAAS_WindNet model also uses the least average number of parameters among all four models. As we can see, in some cases (e.g., ME-autumn, WF1-summer), IAAS_n models use fewer parameters than the IAAS framework, while in some other cases (e.g., NH-spring, WF2-winter), IAAS_n uses much more number of parameters than the IAAS framework. Hence, we utilize a violin plot to show the number of parameters of the obtained models in each scenario in Fig. 15. Note that we truncate the negative values in Fig. 15 because they are meaningless. As observed in the figure, the proposed IAAS framework achieves the most stable results with the least average number of parameters used in the obtained models among the four scenarios. Furthermore, models using the IAAS_n framework achieve the second most stable results in terms of a number of parameters, which indicates that the selector actor component plays a significant role in reducing the number of parameters of network models. Finally, we use a bubble plot to showcase the overall results of the ablation experiments in Fig. 16. We only utilize RMSE results to represent the model forecasting accuracies in Fig. 16 since the plot using MAE vs. model parameter amounts is similar to Fig. 16. As observed, we have the following findings: (i) Comparing IAAS with IAAS_s, selector actor is critically essential in reducing the model structural complexity and obtaining a network with much fewer parameters. (ii) Comparing IAAS with IAAS_n, net pool is paramount in improving the model forecasting accuracy and reducing the model structural complexity at a certain level. (iii) Comparing IAAS with IAAS_sn, our proposed framework is able to not only significantly enhance the model forecasting accuracy but also achieve a network with
simple structures.

Figure 15. Parameter amounts of obtained models using four different frameworks in ablation experiments.

Figure 16. Relationship illustration between model forecasting accuracy and model complexity in ablation experiments. Note: the area of the circle indicates the parameter amounts of the obtained models.

Table 9. Comparisons of the number of parameters of the obtained load forecasting models

|                | IAAS_LoadNet | IAAS_s_LoadNet | IAAS_n_LoadNet | IAAS_sn_LoadNet |
|----------------|--------------|----------------|---------------|-----------------|
| ME-spring      | 3021         | 230840         | 88473         | 157708          |
| ME-summer      | 2141         | 370524         | 30639         | 97873           |
| ME-autumn      | 1117         | 234387         | **977**       | 374648          |
| ME-winter      | 3985         | 147610         | 2277          | 36617           |
| NH-spring      | **29279**    | 174674         | 59885         | 16839           |
| NH-summer      | 785          | 577936         | 1561          | 49046           |
| NH-autumn      | 759          | 149880         | 1231          | 35771           |
| NH-winter      | **1634**     | 294560         | 2935          | 6823            |
### Table 10. Comparisons of the number of parameters of the obtained wind power forecasting models

| Model       | IAAS_WindNet | IAAS_s_WindNet | IAAS_n_WindNet | IAAS_sn_WindNet |
|-------------|--------------|----------------|----------------|-----------------|
| WF1-spring  | 847          | 13061          | 953            | 104184          |
| WF1-summer  | 9533         | 61323          | 2557           | 267298          |
| WF1-autumn  | 653          | 39907          | 1695           | 439208          |
| WF1-winter  | 1787         | 92113          | 1627           | 91409           |
| WF2-spring  | 677          | 27732          | 1019           | 20225           |
| WF2-summer  | 2453         | 18927          | 985            | 6923            |
| WF2-autumn  | 1449         | 61319          | 2085           | 274333          |
| WF2-winter  | 3133         | 119833         | 30287          | 86840           |
| Average     | **2566.5**   | 54276.875      | 5151           | 161302.5        |

6. Discussion and 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 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 the forecasting models. Even though 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 datasets automatically. Even though 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 proposed IAAS framework builds on and significantly extends prior works in function-preserving network transformation operation (Chen et al. 2016) and network transformation control (Cai et al. 2018). Specifically, we first introduce the feature-learning operation in Section 2 by classifying it into two phases: feature extraction and feature generation. With this, we provide the theoretical function-preserving network transformation for RNN in order to better process the time-
series electricity data under the Net2Net framework. Compared to CNN and FCN, RNN is better suited to address the temporal data; therefore, such a theoretical contribution bridges the knowledge gap in a broader application scope of the Net2Net framework. Furthermore, considering the limitations of the EAS framework (Cai et al. 2018) in the control process of the network transformation, such as the manual settings for the search strategy and the local optimal state of the obtained neural architectures, we develop two components, namely selector actor and net pool. In addition, we modify the two meta controllers to overcome the challenges of using the diverse types of network structures. In a nutshell, the proposed IAAS framework has four advantages. First, it allows to better handle the time-series data after introducing the RNN function-preserving network transformations. Second, it provides the option of building a robust neural architecture with diverse types of network structures. Third, it is more intelligent due to introducing a selector actor, which is likely to determine whether to widen, deepen the network or keep the network unchanged appropriately. Fourth, this framework is more effective due to introducing a net pool component, which is designed to improve the forecasting accuracy by avoiding the local optimal state of the searched neural architecture.

The experiments have been conducted on two types of electricity energy data, i.e., load and wind power data. Each type of data contains two different datasets. Due to the seasonal patterns of the energy data, each dataset is divided into four subsets, and there are 16 subsets in total. The experimental results demonstrate that the proposed IAAS framework predominately outperforms the ten existing baseline models (or methods) in the average RMSE and MAE forecasting accuracy results. Moreover, the proposed IAAS framework has competitively stable performance in terms of the RMSE and MAE results. Hence, we conclude that the proposed IAAS framework outperforms the existing methods. 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 the selector actor component is probable to reduce the model structural complexity by building a network with much less number of parameters. In contrast, the net pool component is likely to improve the forecasting model accuracy significantly.
In a practical viewpoint, our proposed framework can be incorporated into a moving-window algorithm directly so that the forecasting model can frequently adjust its architectures for adapting the latest changes in the electricity data. 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. Finally, 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.

Endnotes

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

Declaration of Competing Interest

All authors declare that they have no conflict of interest in this work.

Acknowledgement

This study was supported by the National Natural Science Foundation of China (Grant No. 72101066, 72131005).

Reference

Li, X. R., Zhao, Z. (2006) Evaluation of estimation algorithms part I: incomprehensive measures of performance. IEEE Transactions on Aerospace and Electronic Systems, 42(4), 1340-1358.

Zhang, Z., Wei, X., Zheng, X., Li, Q., Zeng, D. (2021) Detecting product adoption intentions via multiview deep learning. INFORMS Journal on Computing, DOI: https://doi.org/10.1287/ijoc.2021.1083

Hu, Y., Hong, Y. (2021) SHEDR: an end-to-end deep neural event detection and recommendation framework for hyperlocal news using social media. INFORMS Journal on Computing, DOI: https://doi.org/10.1287/ijoc.2021.1112

Bi, X., Adomavicius, G., Li, W., Qu, A. (2022) Improving sales forecasting accuracy: a tensor factorization approach with demand awareness. INFORMS Journal on Computing, DOI: https://doi.org/10.1287/ijoc.2021.1147

Weron, R. (2006) Modeling and forecasting electricity loads and prices: a statistical approach. Wiley Finance Series, John Wiley & Sons: Chichester, UK.

Chen, Y., Zhao, J., Qin, J., Li, H., Zhang, Z. (2021a) A novel pure data-selection framework for day-ahead wind power forecasting. Fundamental Research, DOI:https://doi.org/10.1016/j.fmre.2021.09.011

Lund, H. (2005) Large-scale integration of wind power into different energy systems. Energy, 30(13), 2402-2412.
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.

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

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

International Energy Agency (2021) Renewables 2021 analysis and forecast to 2026. Retrieved at https://iea.blob.core.windows.net/assets/5ae32253-7409-4f9a-a91d-1493f8b9777a/Renewables2021-Analysisandforecastto2026.pdf.

Meinshausen, M., Meinshausen, N., Hare, W., Raper, S. C. B., Frieler, K., Knutti, R., Frame, D. J., Allen, M. R. (2009) Greenhouse-gas emission targets for limiting global warming to 2 °C. *Nature*, 458, 1158-1162.

Munawer, M. E. (2018) Human health and environmental impacts of coal combustion and post-combustion wastes. *Journal of Sustainable Mining*, 17(2), 87-96.

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

Pryor, S. C., Barthelmie, R. J., Bukovsky, M. S., Leung, L. R., Sakaguchi, K. (2020) Climate change impacts on wind power generation. *Nature Reviews Earth & Environment*, 1, 627-643.

Chen, K., Chen, K., Wang, Q., He, Z., Hu, J., & He, J. (2019). Short-term load forecasting with deep residual networks. *IEEE Transactions on Smart Grid*, 10(4), 3943–3952.

Jalali, S. M. J., Ahmadian, S., A., K., Shafie-khah, M., Nahavandi, S., Catalao, J. P. S. (2021a) A novel evolutionary-based deep convolutional neural network model for intelligent load forecasting. *IEEE Transactions on Industrial Informatics*, 17(12), 8243-8253.

Jalali, S. M. J., Osorio, G. J., Ahmadian, S., Lotfi, M., Campos, V. M. A., Shafie-khah, M., Khorasavi, A., Catalao, J. P. S. (2021b) New Hybrid deep neural architectural search-based ensemble reinforcement learning strategy for wind power forecasting. *IEEE Transactions on Industry Applications*, 58(1), 15-27.

Metaxiotis, K., Kagiannas, A., Askounis, D., Psarras, J. (2003) Artificial intelligence in short term electric load forecasting: a state-of-the-art survey for the researcher. *Energy Conversion and Management*, 44(9), 1525-1534.

Ng, J. Y., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R. (2015) Beyond short snippets: deep networks for video classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4694-4702.

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

Hu, Q., Zhang, S., Yu, M., Xie, Z. (2016) Short-term wind speed or power forecasting with heteroscedastic support vector regression. *IEEE Transactions on Sustainable Energy*, 7(1), 241-249.
Liu, D., Sun, K. (2019) Random forest solar power forecast based on classification optimization. *Energy*, 187, 115940.

Islam, B., Baharudin, Z., Raza, M., Nallagownden, P. (2014) Optimization of neural network architecture using genetic algorithm for load forecasting. *2014 5th International Conference on Intelligent and Advanced System (ICIAS)*, June 3-5, Kuala Lumpur, Malaysia.

Chen, N., Qian, Z., Nabney, I. T., Meng, X. (2014) Wind power forecasts using Gaussian processes and numerical weather prediction. *IEEE Transactions on Power System*, 29(2), 656-665.

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

Shepero, M., van der Meer, D., Munkhammar, J., Widen, J. (2018) Residential probabilistic load forecasting: a method using Gaussian process designed for electric load data. *Applied Energy*, 218, 159-172.

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

Zhou, S., Li, X., Chen, Y., Chandrasekaran, S. T., Sanyal, A. (2021) Temporal-coded deep spiking neural network with easy training and robust performance. *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 35(12), 11143-11151.

Liu, J., Hou, Q., Liu, Z., Chen, M. (2022) PoolNet+: Exploring the potential of pooling for salient object detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, DOI: 10.1109/TPAMI.2021.3140168

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 Research*, 206, 107776.

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

Heo, J., Song, K., Han, S., Lee, D-E. (2021) Multi-channel convolutional neural network for integration of meteorological and geographical features in solar power forecasting. *Applied Energy*, 259, 117083.

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.

Elsken, T., Metzen, J. H., Hutter, F. (2019) Neural Architecture Search: A Survey. *Journal of Machine Learning Research*, 20: 1-21.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., & Wang, X. (2022). A comprehensive survey of neural architecture search: challenges and solutions. ACM Computing Surveys, 54(4), 1-34.

Mendis, H. R., Kang, C-K., Hsiu, P. (2021) Intermittent-aware neural architecture search. *ACM Transactions on Embedded Computing Systems*, 20(5s), 1-27.

Cheng, A-C. Lin, C. H., Juan, D-C., Wei, W., Sun, M. (2020) InstaNAS: instance-aware neural architecture search. *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 3577-3584.
Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q. V. (2019) MnasNet: platform-aware neural architecture search for mobile. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2820-2828.

Khodayar, M., Kaynak, O., Khodayar, M. E. (2017) Rough deep neural architecture for short-term wind speed forecasting. IEEE Transactions on Industrial Informatics, 13(6), 2770-2779.

Pawlak, Z. (1982) Rough sets. International Journal of Computing and Information Sciences, 11(5), 341-356.

Torres, J. F., Gutierrez-Aviles, D., Troncoso, A., Martinez-Alvarez, F. (2019) Random hyper-parameter search-based deep neural network for power consumption forecasting. International Work-Conference on Artificial Neural Networks: Advances in Computational Intelligence, 259-269.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., & Kurakin, A. (2017). Large-Scale Evolution of Image Classifiers. Proceedings of the 34th International Conference on Machine Learning (ICML), 2902–2911.

Xie, L., Yuille, A. (2017). Genetic CNN. 2017 IEEE International Conference on Computer Vision. 1379–1388.

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

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 Research, 1(4), 472-481.

Chen, T., Goodfellow, I., Shlens, J. (2016). Net2Net: Accelerating Learning via Knowledge Transfer. International Conference of Learning Representations.

Baker, B., Gupta, O., Naik, N., & Raskar, R. (2017). Designing neural network architectures using reinforcement learning. International Conference of Learning Representations.

Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. International Conference of Learning Representations.

Zoph, B., Vasudevan, V., Shlens, J., Le, Q. V. (2018) Learning transferable architectures for scalable image recognition. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 8697-8710.

DiPietro, R., Hager, G. D. (2020) Deep learning: RNNs and LSTM. Handbook of Medical Image Computing and Computer Assisted Intervention. The Elsevier and MICCAI Society Book Series, Chapter 21, 503-519.

Cai, H., Chen, T., Zhang, W., Yu, Y., Wang, J. (2018) Efficient architecture search by network transformation. The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), 2787-2794.

Zhong, Z., Yang, Z. Deng, B., Yan, J., Wu, W., Shao, J., Liu, C-L. (2018) BlockQNN: Efficient block-wise neural network architecture generation. arXiv:1808.05584.

Shih, S-Y., Sun, F-K., Lee, H-Y. (2019) Temporal pattern attention for multivariate time series forecasting. Machine Learning, 108, 1421-1441.
Oreshkin, B. N., Amini, A., Coyle, L., Coates, M. (2021) FCGAGA: Fully connected gated graph architecture for spatio-temporal traffic forecasting. *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 9233–9241.

Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C. (2019) Graph WaveNet for deep spatial-temporal graph modelling. *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1907–1913.

Zheng, C., Fan, X., Wang, C., Qi, J. (2020) GMAN: A graph multi-attention network for traffic prediction. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 1234–1241.

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. *Proceedings of the 30th Web Conference*, 1846-1855.

Chen, D., Chen, L., Shang, Z., Zhang, Y., Wen, B., Yang, C. (2021b) Scale-aware neural architecture search for multivariate time series forecasting. *arXiv:2112.07459*.

Dong, D., Sheng, Z., Yang, T. (2018) Wind power prediction based on recurrent neural network with long-short term memory units. *2018 International Conference on Renewable Energy and Power Engineering (REPE)*, 24-26 Nov. 2018, Toronto, On, Canada.

Kisvari, A., Lin, Z., Liu, X. (2021) Wind power forecasting-a data-driven method along with gated recurrent neural network. *Renewable Energy*, 163, 1895-1909.

Yahya M. A., Hadi, S. P., Putranto, L. M. (2018) Short-term electric load forecasting using recurrent neural network (study case of load forecasting in central java and special region of Yogyakarta). *2018 4th International Conference on Science and Technology (ICST)*, 7-8 Aug. 2018, Yogyakarta, Indonesia.

Samuel, D. (2019) A thorough review on the current advance of neural network structures. *Annual Reviews in Control*, 14, 200-230.

Ahmed, T. (2018) Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2), 334-340.

Bahdanau, D., Cho, K., Bengio, Y. (2015) Neural machine translation by jointly learning to align and translate. *International Conference of Learning Representations (ICLR)*.

Williams, R. J. (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 229-256.

Simonyan, K., Zisserman, A. (2015) Very deep convolutional networks for large-scale image recognition. *International Conference of Learning Representations*.

Zhang, Y., Li, Y., Zhang, G. (2020) Short-term wind power forecasting approach based on Seq2Seq model using NWP data. *Energy*, 213, 118371.

Kingma, D. P., Ba, J. (2017) Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Passalis, N., Tefas, A., Kannaiinen, J., Gabbouj, M., Losifidis, A. (2020) Deep adaptive input normalization for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*. 31(9), 3760-3765.

Cevik, H. H., Cunkas, M., Polat, K. (2019) A new multistage short-term wind power forecast model using decomposition and artificial intelligence methods. *Physica A*, 534, 122177.
He, F., Liu, T., Tao, D. (2019) Control batch size and learning rate to generalize well: theoretical and empirical evidence. 33rd Conference on Neural Information Processing Systems (NeurIPS), Vancouver, Canada.

Rabier, F. (2005) Overview of global data assimilation developments in numerical weather prediction centres. Quarterly Journal of the Royal Meteorological Society, 131(613), 3215-3233.