

Deep Learning Techniques for Power System Operation: Modeling and Implementation

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Yan Du

August 2020

Acknowledgement

This dissertation was completed under the guidance of my Ph.D. advisor, Dr. Fangxing (Fran) Li. I would like to express my sincerest gratitude to Dr. Li for his consistent and patient mentoring through my years of Ph.D. studies. I'm deeply impressed by his erudition and expertise, as well as his amiable and humorous nature. I attribute much of my academic progress and professional development to Dr. Li, for his wise instructions and inspiring encouragement. I regard working with Dr. Li as a great honor, and my Ph.D. study experience at UTK will remain a cherished treasure during the rest of my life.

My special thanks go to my committee members, Dr. Yilu Liu, Dr. Hairong Qi, and Dr. James Ostrowski for devoting time to my dissertation defense and for their instructive suggestions to improve the quality of my dissertation.

I would also like to thank my group mates, Dr. Hantao Cui, Dr. Qingxin Shi, Xiao Kou, Wei Feng, Qiwei Zhang, Xiaofei Wang, Haoyuan Sun, Buxin She, Mariana Kamel, Cody Rooks, Evan McKee, and Ian Schomer, for their help during my studies and for all the pleasant time we have spent together in off-hours. I am also grateful for the support from all the faculty, staff and students at the Center for Ultra-Wide-Area Resilient Electric Energy Transmission (CURENT). The openness, inclusiveness and creativity at the center has made the journey of study even more enjoyable.

Finally, my deepest gratitude goes to my parents, for their years of priceless love, care, and support for my study and growth. The invaluable care from my family can never be repaid. I will carry on with their blessings and expectations to write the next chapter of my life.

Abstract

The fast development of deep learning techniques in recent years has drawn attention from both academia and industry. And there have been increasing applications of the DL techniques in many complex real-world problems, including computer vision, medical diagnosis, and natural language processing. The great power and flexibility of deep learning can be attributed to its hierarchical learning structure that automatically extracts features from mass amounts of data, as well as its end-to-end solving mechanism that directly generates the output from the input, which considerably improve the computational efficiency.

The power system is one of the most complex artificial infrastructures, and many power system control and operation problems share features with the real-world applications mentioned above, such as time variability and uncertainty, and partial observability, which impedes the performance of conventional model-based methods. On the other hand, with the wide spread implementation of measuring systems providing massive data from the field, the data-driven deep learning technique is becoming an intriguing alternative method to enable the future development and success of the smart grid.

This dissertation explores the potential of utilizing deep-learning-based approaches to solve a broad range of power system modeling and operation problems. First, a comprehensive literature review summarizing the existing applications of deep learning techniques in power systems is presented. Second, the prospective application of deep learning techniques in several scenarios in power systems, including contingency screening, cascading outage search, multi-microgrid energy management, residential HVAC system control, and electricity market bidding are discussed in detail in the following chapters. The problem formulation, the specific deep learning approaches in use, the simulation results, and comparisons with the model-based methods are all presented. Finally, the conclusions and future directions are provided in the last chapter. It's hoped that this dissertation will work as a single spark that can generate more innovative ideas and original studies,

widening and deepening the application of deep learning techniques in the field of power systems, and eventually bring about some positive impacts on the resilient and economic control and operation of the real-world bulk grid.

Contents

Chapter 1 Introduction.....	1
1.1 Deep Learning: Motivation and Development	1
1.2 Classification and Application of Deep Learning	4
1.3 Applications of Deep Learning in Power Systems	6
1.3.1 Supervised learning in power system control and operation	6
1.3.2 Unsupervised learning in power system control and operation.....	9
1.3.3 Reinforcement learning in power system control and operation	10
1.4 Summary.....	11
Chapter 2 N-1 Contingency Screening with Deep Convolutional Neural Network.....	13
2.1 Introduction	13
2.2 Composite Security Index of Power System Security Assessment	15
2.3 Deep CNN-based N-1 Contingency Screening	17
2.3.1 A brief on deep CNN	17
2.3.2 Mapping power system data to deep CNN input data	18
2.3.3 Constructing deep convolutional neural network	20
2.4 Case Study	25
2.5 Conclusions	28
2.6 Acknowledgement	28
Chapter 3 Fast Cascading Outage Screening based on Deep CNN and Depth-First Search	29
3.1 Introduction	29
3.1.1 Motivation	29
3.1.2 Literature review.....	30
3.1.3 Contributions	32
3.2 Deep CNN-based Security Assessment.....	33
3.2.1 Mapping power grid data to deep CNN input data.....	33
3.2.2 Constructing deep convolutional neural network	35
3.2.3 Training sample generation	38
3.3 Cascading Outage Screening based on Depth-First Search Algorithm	39

3.4 Simulation Analysis	43
3.4.1 Deep CNN regression of ACOPF model	43
3.4.2 Identifying cascading outage path with DFS algorithm	47
3.5 Conclusions	50
3.6 Acknowledgement	51
Chapter 4 Multi-microgrid Energy Management with Deep Learning and Reinforcement Learning.....	52
4.1 Introduction	52
4.2 Modeling of Multi-microgrid Energy Management.....	56
4.3 Multi-microgrid Operation Simulation with Deep Neural Network	59
4.3.1 Deep neural network structure.....	60
4.3.2 DNN training algorithm.....	62
4.4 Monte Carlo Reinforcement Learning Method for DSO Decision-making.....	63
4.4.1 A brief overview of reinforcement learning	64
4.4.2 Mapping multi-microgrid energy management problem to reinforcement learning	65
4.4.3 Model-free Monte Carlo method	66
4.5 Simulation Analysis	68
4.5.1 Simulating multi-microgrid Operation with DNN	69
4.5.2 Monte Carlo method for optimizing DSO pricing strategy	74
4.6 Conclusions	79
4.7 Acknowledgement	80
Chapter 5 Multi-zone Residential HVAC Control using Deep Reinforcement Learning	81
5.1 Introduction	81
5.2 Multi-zone Residential HVAC System Control Problem Formulation	85
5.2.1 A brief introduction of the multi-zone HVAC system control problem.....	85
5.2.2 Mapping HVAC control problem to Markov Decision Process	85
5.3 DDPG-based Control Strategy for Multi-zone HVAC System	88
5.3.1 A brief review of deep reinforcement learning methods	88
5.3.2 Understanding the basic principles behind typical deep RL methods	90
5.3.3 Realizing the continuous control of HVAC system with DDPG	91
5.4 Case Study	95

5.4.1 Simulation environment	95
5.4.2 Design of the DNN structure in Deep RL	96
5.4.3 Performance of the continuous HVAC control method.....	97
5.5 Conclusions	110
5.6 Acknowledgement	111
Chapter 6 Solving Markov Game in Day-ahead Electricity Market with Multi-Agent Deep RL	112
6.1 Introduction	112
6.2 Multi-agent Market Bidding Problem Formulation	115
6.2.1 A brief on electricity market and bidding strategies	115
6.2.2 Mathematical formulation of day-ahead electricity market clearing.....	117
6.2.3 Markov game model of day-ahead electricity market bidding	118
6.3 MADDPG Method for Day-ahead Electricity Market Bidding	120
6.3.1 An overview of reinforcement learning method.....	120
6.3.2 Deep deterministic policy gradient method for continuous control	121
6.3.3 MADDPG for solving Markov game in day-ahead electricity market bidding	122
6.3.4 Baseline cases for evaluating the learning performance of MADDPG	125
6.4 Simulation Analysis.....	125
6.4.1 Test system description.....	125
6.4.2 Design of neural network and simulation platform	127
6.4.3 Nash Equilibrium strategy from MADDPG: uncongested case	128
6.4.4 Solving Markov game with MADDPG: congested case	131
6.5 Conclusions	135
6.6 Acknowledgement	135
Chapter 7 Conclusions and Future Works	136
7.1 Summary of Current Progress	136
7.2 Future Works	138
References	141
Appendix.....	153
Vita	155

List of Tables

Table 2.1. AC power flow results based on deep CNN	26
Table 2.2. Test time comparison	27
Table 2.3. AC power flow results of ANN	27
Table 3.1. Summary of training/test set generation	44
Table 3.2. Sample set size for deep CNN training and testing	44
Table 3.3. ACOPF regression results based on deep CNN.....	45
Table 3.4. Test time comparison	46
Table 3.5. ACOPF regression based on traditional ANN	47
Table 3.6. Time efficiency of DFS algorithm	48
Table 3.7. Cascading outage screening results	49
Table 3.8. Index of most frequently tripped lines in each cascading outage stage.....	50
Table 4.1 Microgrid composition	69
Table 4.2. Parameters of distributed energy resources	70
Table 4.3 Summary of DNN training settings	72
Table 4.4. Computation time for DNN and Monte Carlo method.....	76
Table 5.1. Daily user comfort level	96
Table 5.2. DNN structure applied in DDPG and DQN algorithm	96
Table 5.3. Test results of different HVAC control methods.....	99
Table 5.4. Comparison of optimization results for different building models.....	106
Table 5.5. Test results of different control methods (under PJM price)	106
Table 6.1 GENCO generation cost function.....	127
Table 6.2. Design of DNNs in deep RL.....	127
Table 6.3. Comparison of MADDPG with baseline: uncongested case.....	131
Table 6.4 Comparison of MADDPG with baselines: congested case.....	132

List of Figures

Figure 1.1. Relationships of deep learning to machine learning and AI technology.....	1
Figure 1.2. Output from each layer in a well-trained convnet.....	3
Figure 2.1. The principle of deep CNN feature extraction	18
Figure 2.2. Illustration of convolution operation.....	21
Figure 2.3. Deep CNN structure for N-1 contingency screening	25
Figure 3.1. Deep CNN structure for security assessment.....	37
Figure 3.2. Flowchart of generating cascading outage training samples.....	39
Figure 3.3. Multi-scenario tree for cascading outage screening.....	40
Figure 3.4. Flow chart of depth-first search algorithm.....	42
Figure 3.5. ANN structure for security assessment	46
Figure 4.1. Multi-microgrid energy management under DSO pricing control.....	57
Figure 4.2. Multi-layer structure of the Deep Neural Network	60
Figure 4.3. Illustration of reinforcement learning	65
Figure 4.4. Simulation framework for multi-microgrid energy management	72
Figure 4.5. Disturbance of retail price	74
Figure 4.6. Disturbance of microgrid load	74
Figure 4.7. Optimal price setting under different weight factors	76
Figure 4.8. Total profit and final PAR under different weight factors.....	77
Figure 4.9. Comparison of Monte Carlo method and intuitive method	78
Figure 5.1. DNN structure for function approximation in RL.....	89
Figure 5.2. Multi-zone HVAC control framework with DDPG	95
Figure 5.3. Convergence of different deep RL methods.....	98
Figure 5.4. Setpoint control strategy based on DDPG for 10 test days (top: zone 1; bottom: zone2)	101

Figure 5.5. Setpoint control strategy based on DQN for 10 test days (top: zone 1; bottom: zone2)	102
Figure 5.6. Setpoint control strategy from the rule-based case for 10 test days (top: zone 1; bottom: zone2)	103
Figure 5.7. Setpoint control strategy from the fixed setpoint case for 10 test days (top: zone 1; bottom: zone2)	104
Figure 5.8. Illustration of the comparison results	105
Figure 5.9. Setpoint control strategy based on DDPG under PJM price for 10 test days (top: zone 1; bottom: zone2)	108
Figure 5.10. Setpoint control strategy based on DQN under PJM price for 10 test days (top: zone 1; bottom: zoomed part of zone 1)	109
Figure 5.11. Setpoint control strategy in the fixed setpoint case under PJM price for 10 test days (top: zone 1; bottom: zoomed part of zone 1)	110
Figure 6.1 MADDPG algorithm for solving Markov game in day-ahead electricity market	126
Figure 6.2. Topology of the IEEE 30-bus system	126
Figure 6.3. Load profile in June and July of year 2019	130
Figure 6.4. Convergence of MADDPG in the uncongested case	130
Figure 6.5. Bidding strategy of three GENCOs with MADDPG: uncongested case	130
Figure 6.6. Convergence of MADDPG in the congested case	131
Figure 6.7. Bidding strategy of three GENCOs with MADDPG: congested case	132
Figure 6.8. Hourly market clearing price under MADDPG bidding strategy	132
Figure 6.9. Convergence of DQN in the congested case	134
Figure 6.10. Bidding strategy of three GENCOs with DQN: congested case	134

Chapter 1 Introduction

This chapter gives a brief introduction to deep learning techniques and their various applications in both academia and industry, specifically in the area of power systems.

1.1 Deep Learning: Motivation and Development

The debut of AlphaGo in year 2016 drew worldwide attention to the evolutionary deep learning (DL) technique, which has continued developing at a fantastic speed [1]. Deep learning is a subset of machine learning. The relationship of deep learning to machine learning and AI technology is shown in Figure 1.1 [2]:

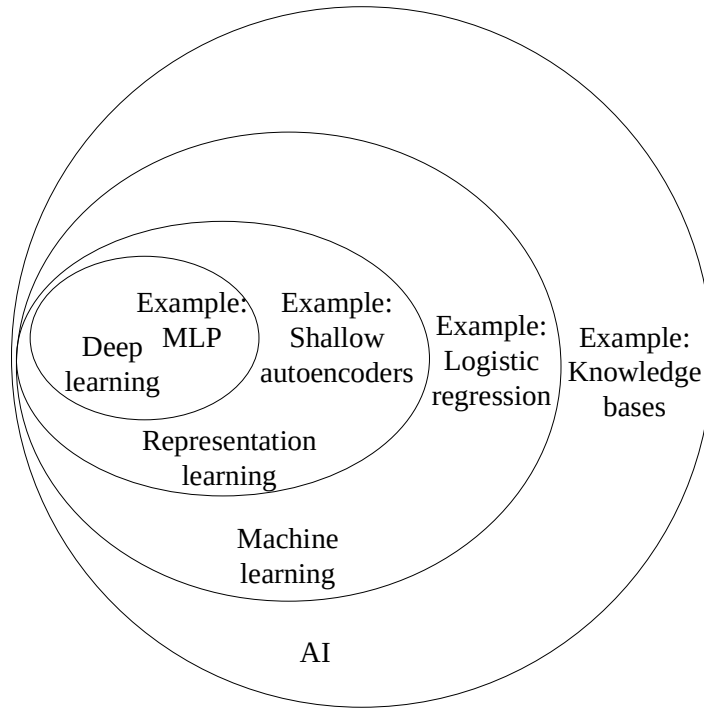


Figure 1.1. Relationships of deep learning to machine learning and AI technology

The origins of deep learning can be dated back to the 1940s. However, deep learning has only recently caught the public attention because it remained unpopular during the time it was proposed, and also because it has gone through many different names before it is finally called “deep learning”, which started in 2006 [2]. The core idea of deep learning is the successive layers of representation. A representation means a way to encode the data. For example, a color image can be represented by the RGB matrices; a figure can be represented by its binary format. Deep learning intends to find a meaningful representation of the input data so that the expected output can be achieved [3]. The term “deep” refers to the multiple layers that are connected end to end to learn the data representations. The idea of multi-layer representation is based on the assumption that the data in the real-world can all be regarded as composition of features. The authors in [4] have developed sophisticated experiments and detailed explanations for how multiple layers work in a hierarchical way to capture local features and gradually form the high-level concept, and also a vivid visualization example of the output from each layer, as shown in Figure 1.2 [4]:

The goal of the convnet model in Figure 1.2 is to classify a large number of 2-D color images based on their contents. As shown in the figure, the output from the first layer are simply edges and colors; the output from the second layer begins to show corners and other edge/color conjunctions; the output from the third layer has more invariance and captures similar textures; layer 4 begins to show class-specific features like the dog faces and bird legs; finally, layer 5 outputs the entire objects with significant pose variation, like the dogs and keyboards, which is obvious enough for the computer to differentiate the various types of the objects. In summary, the shallow layers first capture the more general and simple local patterns such as lines and shapes. As the layers go deeper, features from the previous layers will be combined and form larger and more complicated patterns that are closer to the expected output.



Figure 1.2. Output from each layer in a well-trained convnet

The multiple layers for data representation are embedded within the neural network models, which are almost always used in the deep learning studies. In the traditional machine learning methods, the neural network usually only has one hidden layer. This simple structure leads to two major challenges that inflicts the generalization of the traditional machine learning algorithms [2]. The first challenge is the curse of dimensionality, which means that the input data has multiple dimensions, and the traditional machine learning method cannot fully capture the spatial or temporal correlations crossing the different dimensions with its limited representation ability. In

addition, many machine learning algorithms are built upon prior beliefs that the target function has smoothness or local constancy, which means the function does not change much within a small region. This is a very strict assumption and it does not apply to statistical challenges involved in solving AI-level tasks. The deep learning method, however, replaces this assumption with a much milder one that the data to learn is generated by the composition of features, from simple to abstract, following a hierarchical structure, as has been mentioned above. The relaxed assumption allows the deep learning method to fit to more complex and high-dimensional functions, and to obtain better generalization than the traditional machine learning method.

Today, with the improvement of the computer hardware and software infrastructure, as well as the accessibility to massive amount of data as training samples, it becomes possible to build and train neural networks with growing depth for solving increasingly complicated real-world problems, and their accuracy improves over time as available computation resources increase. Some current exemplary breakthroughs of the deep learning technique include near-human level image classification/speech recognition/autonomous driving, superhuman go playing, etc.[3]. With the explosion of data and the continuous advancement of the hardware and cloud-based computation resources, deep learning technology will undoubtedly be applied across more and more potential fields and serve societal well being in this information era.

1.2 Classification and Application of Deep Learning

There are three important branches in the field of machine learning: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is a type of machine learning with labeled data as training samples. The algorithm aims to formulate a mapping between the input and the output based on the large numbers of correct samples. The labeled samples can be regarded as the “supervisor” to provide guidelines for tuning the parameters of the learning model and to lead it toward the more accurate formulation. There are two main subgroups of supervised learning, the classification and the regression. With regard to the former, computer

vision, handwriting recognition, and medical diagnosis are its real-world realizations. As for the latter, regression is the problem of estimating or predicting a continuous quantity. Stock price forecasting, speech recognition, and machine translation are all in essence a supervised regression learning process.

Unsupervised learning is a completely different class from supervised learning. In unsupervised learning, there are no labeled samples as the correct answers for the machine to learn. A typical example of unsupervised learning is the clustering problem, such as grouping customers with similar purchasing inclinations, identifying fake news and spam emails, and document classification based on tags and contents. Another example of unsupervised learning is the autoencoder. An autoencoder consists of two parts, the encoder and the decoder. The encoder develops a proper way to represent the input data and the decoder transforms the representation back to the input space. The autoencoder belongs to unsupervised learning because there is no labeled data used during the learning and the neural network only builds reconstructions of the input. Some useful applications of autoencoder include data denoising and dimensionality reduction, where the autoencoder extracts the most meaningful features from the input data and formulate a compact representation.

Reinforcement learning is a third type of machine learning problem that aims at optimizing the time-sequential decision strategies via learning. Unlike the above two types of learning, reinforcement learning does not rely on large quantities of data, and its performance is evaluated by a reward signal. In reinforcement learning, an agent is placed in an unknown environment. At each time step, the agent takes an action, and is transferred to the next state following a transition probability. The agent will get a reward from the environment as feedback of taking a certain action. The goal of the agent is to maximize the total reward after reaching the end state.

There have been a broad range of applications of reinforcement learning, from playing video games to training self-driving cars. The independence of reinforcement learning from exact models and data gives it high flexibility to adapt to problems that are unobservable or partially observable,

or have enormous or infinite solution spaces.

All three types of machine learning methods have been revitalized in combination with the deep neural network, which is the hard core behind the deep learning technique. In the field of power systems, studies applying the deep learning techniques for power system control and operation have just begun, but they have already covered a wide range of topics. In the following subsections, a comprehensive review of the most recent related research works will be presented.

1.3 Applications of Deep Learning in Power Systems

The power system is a highly complex, multi-dimensional artificial infrastructure that shares many features with the above-mentioned real-world tasks, such as time variability, partial observability, and random uncertainty. Conventional model-based methods encounter difficulty when analyzing the stable and transient operation features of power systems, especially with the increasing penetration of renewable energy, demand response resources, as well as the fusion with the information, communication, and transportation networks. The deep learning technique provides a brand-new way to overcome the issue with high-dimensional data mining and feature extraction, and to compensate for the training data insufficiency and low generalization ability that inflicts the traditional shallow machine learning methods. The following sections present the existing power system researches in all the three machine learning categories.

1.3.1 Supervised learning in power system control and operation

1) Forecasting

One major application of supervised learning in power systems is the forecasting, where large quantities of historical data are available for algorithm training. The newly developed deep neural network can capture the temporal or spatial correlations between the inputs with its hierarchical structure, and to provide a more accurate forecast result for time series data.

The most commonly used deep neural network in forecasting is called the long-short-term-memory (LSTM) recurrent neural network. It is different from the conventional feedforward neural

network due to the fact that it not only has connections between different layers, but also has connections between neurons in the same layer, which makes it possible to capture the temporal correlations of the inputs at different time steps, and an ideal fit for time sequential data. Ref. [5]-[6] apply the LSTM for short-term residential load forecasting, and they compare it with a series of traditional machine learning methods including space vector machine (SVM) and K nearest neighbor (KNN) to demonstrate the improved accuracy of the method. Ref. [7] further considers the impact of fluctuated PV generation on the net load forecast, and proposes a Bayesian deep learning, which combines Bayesian probability theory with the LSTM. Instead of outputting the deterministic net load forecast, the model generates the estimated probability distribution of the net load to cover uncertain scenarios. Authors in [8]-[9] focus on short-term wind speed forecasting, where the LSTM is applied to capture the temporal features of the wind data, and convolutional neural network (CNN) is applied to capture the spatial features of the wind data. The combination of the two methods considerably improves the forecast accuracy. Ref. [10] proposes a bidirectional LSTM to forecast day-ahead load, wind and PV to help virtual power plants to optimize their bidding strategy in the day-ahead wholesale market. Apart from LSTM, other deep neural networks, such as deep residual networks [11] and quantile regression neural network [12] have also been used for load forecasting.

2) Load monitoring and identification.

Another important application of supervised learning in power systems is load monitoring and load identification. Load monitoring is a technology to disaggregate the cumulative energy consumption of a customer into appliance-level consumptions. Load identification is to analyze some characteristics of customer energy consumption based on the smart meter data. Both studies make way for more advanced smart grid applications such as load forecast for individuals, customized demand response programs, and energy-efficient appliance utilization. In [13], the deep convolutional neural network (deep CNN) is designed as a classifier to detect Type II appliances, i.e., the appliances that have multiple operation modes, such as washing machines and

dish washers. Deep CNN is applied to capture the local dependencies of the appliance power consumption patterns. In [14], the authors further consider the simultaneous detection of multiple appliances, and they propose a deep dictionary learning method to solve the multi-label classification. In [15], the LSTM neural network is applied to identify the parameters of the time-varying ZIP load model and time-varying induction motor model under the random impacts of weather conditions and customer behaviors. In [16], the authors apply deep CNN to extract features from massive load profiles and use a support vector machine to identify socio-demographic information of the customers from the extracted features, such as age and social class. In [17], a wide and deep CNN is designed to capture the periodicity of the customer electricity consumption and to further detect the potential electricity theft.

3) Security assessment and fault diagnosis

Supervised learning has also been adopted as a monitoring tool to identify the power system potential vulnerability and risks for the sake of safe and reliable operation. In [20], a hierarchical deep domain adaptation (HDDA) approach is proposed as a system fault classifier. With its layered feature learning framework, the well-trained HDDA model can be transferred to detect faults under different loading conditions, which cracks the obstacle of training data insufficiency. In [21], the deep convolutional neural network and PMU measurements are combined for faulted line localization and transient stability monitoring, respectively. In [22], the deep neural network is applied for real-time event classification under renewable penetration. The deep autoencoder is applied in [23]-[25] for system security assessment and system islanding detection, due to its automatic feature extraction ability. In [26], the deep CNN is applied to predict the transient stability to provide an early termination for time-domain simulation based on existing simulation results.

4) Other applications

Some other applications of supervised learning include optimizing the charging schedule of electric vehicles [18], predicting AGC control signal [19], identifying false data injection [27], and

state estimation [28], etc.

In summary, data acquisition is key to supervised learning. For problems where massive amounts of data are easily accessible, such as the forecast problem, deep learning methods outshine traditional machine learning methods because their multi-layer hierarchical structures allow for more delicate feature extraction and more complex model regression. However, in some real-world scenarios, when the training data is not readily available, the performance of supervised learning can be seriously limited. In such cases, the deep reinforcement learning, introduced in the following section, will work more efficiently.

1.3.2 Unsupervised learning in power system control and operation

As has mentioned before, one representative unsupervised learning model is the deep autoencoder (DAE). The learning process of the DAE usually consists of two stages: the unsupervised pre-training stage and the supervised fine-tuning stage. In the pre-training stage, a DAE is trained to extract features from the input for reconstruction. The goal at this stage is to generate the output that is as close as possible to the input data with nonredundant features. The pre-training provides a good network parameter initialization and avoids the model to get stuck in the local optimum. In the fine-tuning stage, only the well-trained encoder part of a DAE is kept to efficiently extract features from the labeled data, and the network parameters are further updated via backpropagation until the target output is achieved.

In [30]-[34], DAE is applied for forecasting uncertain factors including electricity price, wind generation, and solar irradiance. In [29], DAE is applied as a cluster tool to categorize a vast number of daily load profiles to better analyze customer responsiveness to dynamic price signals. In [35], DAE is proposed as a novel cyber attack tool, where it is trained using the normal measurements, and the anomalies will result in large reconstruction error. In [36], a Monte Carlo tree search method is applied for fast generator start-up after large-scale blackout, where DAE is built as the value network to evaluate the current generator start-up scheme at each search step.

Generative adversarial networks (GANs) is another representative of unsupervised learning. As

can be inferred from the name, GAN is mostly applied for generating data samples. This function is especially desired in the cases when there is no easy access to available historical data, or when generating the training data using the conventional analytic method is too costly. GAN consists of two deep neural networks, the generator and the discriminator. The function of the generator is to produce data samples that follows the distribution of the historical data, and the function of the discriminator is to discriminate the generated data from the actual historical data. By training the two neural networks simultaneously, the model will eventually reach an equilibrium where the discriminator can no longer tell the generated data from the actual data, which means that the generator can produce data that is realistic enough for further analysis. Authors in [37] propose to apply GAN to create renewable scenarios to cover the full diversity of the uncertainties. The generated scenarios can be used for further power system planning and operation. In [38], GAN is applied to make up for missing PMU measurement data for dynamic security assessment.

1.3.3 Reinforcement learning in power system control and operation

Reinforcement learning utilizes a reward mechanism instead of the labeled samples to guide the learning behavior. In power system studies, reinforcement learning has been applied to a wide range of time-sequential optimal decision-making problems. Deep reinforcement learning (deep RL) is a combination of the deep neural network (DNN) with reinforcement learning. The DNN is utilized to estimate the Q value or the possibility for each state-action pair during the learning. In the conventional reinforcement learning, a tabular method is applied, where all the states and actions are listed as a 2-D table, and the Q-values of each pair are filled in. The tabular method can only be applied for discrete action space. And once the state space changes, the table needs to be rebuilt. By contrast, the data-driven DNN gives deep RL high generalization to adapt to new environments. There are also certain types of deep RL methods that can deal with continuous action spaces, which will be introduced in the following section.

In [39], a deep Q learning method is developed to control a cluster of thermostatically controlled loads (TCLs), where the deep CNN is utilized to estimate the Q value of the on/off action of the

TCL. The simulation results show that the proposed method can reduce the electricity cost based merely on the observation of the air temperatures of TCLs. In [40], both the deep Q network (DQN) and the deep policy gradient (DPG) methods are applied to optimize the scheduling of residential loads including air conditioners, electrical vehicles, and dishwashers. The difference between the two methods is that the former estimates the Q value of the action, while the latter estimates the probability of the action. Both methods have higher scalability to adapt to larger state spaces than the conventional heuristic method due to the generalization of the DNN. In [41]-[42], deep Q learning is applied to decide the daily optimal operation schedule of the microgrid with multiple distributed generators and energy storages. In [43], deep Q network is applied to optimize the charging schedule of electric vehicles with the combination of LSTM for predicting price signals as the input state. In [44], a deep deterministic policy gradient (DDPG) method is applied to optimize the joint bidding strategy of the load serving entity in both the wholesale and retail market. The advantage of DDPG over DPG and DQN is that the former can deal with a continuous action space. In [45], DDPG is applied to determine the generation command to maintain a steady local frequency.

1.4 Summary

In this chapter, a detailed overview of the existing researches regarding deep learning techniques in power systems is provided to give the readers a conceptual perception of the tremendous potentials of deep learning in solving complicated real-world problems, both theoretically and practically. In the next several chapters, some of the most complex power system operation problems are presented, along with the initial attempts of applying deep learning techniques to solving these problems. The feasibility of applying deep learning in real-world applications will also be discussed.

In Chapter 2, a novel data-driven contingency screening method for power system operation under uncertain scenarios is introduced, which is based on deep CNN. Following that, in Chapter

3, a fast cascading outage screening method is proposed, which considers the sequential spread of the outages within the entire system. The method is a combination of both deep CNN and depth-first-search (DFS) method, where the former is utilized to estimate the system security status, and the latter is utilized to identify the contingency path with highest severity. The method is also compared with conventional model-based power flow methods to verify its accuracy and computational efficiency. In Chapter 4, a model-free RL method combined with DNN is presented for realizing the economic energy management of multi-microgrid system in connection to the distribution system. In Chapter 5, the DDPG method is applied to optimize the setpoint of a multi-zone residential HVAC system. In Chapter 6, a multi-agent DDPG method is employed for solving the Markov game at the day-ahead electricity market to optimize the bidding strategies of each generation company (GENCO) bidder. Finally, Chapter 7 summarizes the current researches and provides directions for future works.

Chapter 2 N-1 Contingency Screening with Deep Convolutional Neural Network

The increasing penetration of renewable energy makes the traditional N-1 contingency screening highly challenging when a large number of uncertain scenarios need to be combined with contingency screening. In this chapter, a novel data-driven method, which is similar to the image-processing technique, is proposed for accelerating N-1 contingency screening of power systems based on the deep convolutional neural network (CNN) method. Once the deep CNN is well trained, it has high generalization and works in a nearly computation-free fashion for unseen instances such as topological changes in the N-1 cases and uncertain renewable scenarios. The proposed deep CNN is implemented on several standard IEEE test systems to verify its accuracy and computational efficiency. The proposed study constitutes a solid demonstration of the considerable potential of the data-driven deep CNN in future online applications.

2.1 Introduction

The increasing penetration of renewable energy into the bulk power system has brought the issue of uncertainty, which leads to higher requirement on system operation security. Security assessment decides whether the system is operating safely, critically, or unsafely based on a series of criteria including voltage level, power flows, islanding, etc. [46] Security assessment can be used as a reference for system operators to take preventive measures against operation risks.

The N-1 contingency screening is a crucial part of security assessment. The N-1 contingency refers to the loss of any single element, e.g., a transmission line or a generator, in the power system. The main challenge for N-1 contingency screening under uncertainty is the extreme model complexity in the case of large-scale power systems, combined with many uncertain scenarios. For instance, a traditional contingency screening for an N-branch system requires N power flow runs; however, once this is combined with M independent wind plants with 10 uncertain scenarios in

each plant, contingency screening needs to be performed for $N \times 10^M$ power flow runs [47]. Even though it is well known that a traditional full-fledged N-1 contingency screening in the actual ISO operation takes only tens of seconds to complete, it will be unmanageable if many wind scenarios must be combined.

Tremendous research efforts have been dedicated to accelerating computation speed for N-1 contingency screening in the literature. In [48], a mixed integer linear programming model is formulated to calculate system reserve margin with renewable and load uncertainties involved. To reduce model complexity for large-scale power systems, a fast security assessment approach is proposed in [49] by removing redundant constraints from the original model, while keeping the same feasible region. Similarly, in [50], an iterative methodology is proposed for filtering only the active N-1 congestion constraints with the utilization of a line outage distribution factor in order to reduce the computational burden. A multi-level filtering algorithm for operation scenario selection is proposed in [51] to decrease the number of constraints in stochastic transmission planning with N-1 contingency analysis.

All of the above methods can be summarized as the model-based method, where a large set of algebraic equations needs to be solved for security assessment. However, the model-based method faces the problem of identification inaccuracy and computational inefficiency, which constitutes a major impediment for its online application. In contrast to the model-based method, the data-driven approach relies on raw data for direct system analysis to avoid identification error, and with generalization to unseen inputs, it reduces computational burden. The above features make the data-driven method a fast and reliable alternative tool for security assessment in real-time scenarios.

In terms of the data-driven approach for security assessment, some machine learning methods, including decision tree, artificial neural network (ANN), and support vector machine, have been introduced in [52]-[54]. In these studies, the algorithm relies on state variables, e.g., voltage magnitude and voltage angle, to quantify system security level. In such cases, the power flow needs to be calculated first to obtain the required state variables, which can be computationally costly in

the case of large-scale systems with multiple scenarios involved.

To address the above issues with both the model-based and data-driven method for security assessment, a novel data-driven approach for static security assessment with N-1 contingency based on deep CNN is presented in this chapter. If compared with other data-driven approaches, the major highlight of the method is that the proposed deep CNN only depends on the known parameters, i.e. system topology and bus power injection, instead of system state variables, for evaluating system security status. Hence, once the model is well-trained, it can be readily applied to new test cases with little computation effort. This nearly computation-free feature of deep CNN makes it a desirable tool for online applications.

The rest of the chapter is organized as follows: section 2.2 briefly introduces the composite security index for system security assessment; section 2.3 explains the basic idea of deep convolutional neural network and the design of the proposed deep CNN model; section 2.4 verifies the deep CNN performance on IEEE standard test cases; finally, section 2.5 concludes the chapter.

2.2 Composite Security Index of Power System Security Assessment

To accurately evaluate the security status of the power system under N-1 contingency, a composite security index is first introduced, which measures both bus voltage limit violation and line flow violation. For each measurement, two types of limits are defined, the security limit and the alarm limit. Security limit refers to the maximum allowed range for the bus voltage and line flow, and alarm limit indicates the closeness of the system to the limit violation. Accordingly, the system security status can be categorized into three types: secure, alarm, and insecure. A system is in the alarm state if at least one of the measurements violates the alarm limit but is still within the security limit. A system is insecure if at least one of the measurements violates the security limit [55]. Several other measures need to be defined before proceeding to calculate the composite security index.

For bus voltage, the normalized deviation of bus voltage from the alarm limits is defined as

follows:

$$d_{v,i}^u = \begin{cases} \frac{|v_i - A_i^u|}{V_i^b}, & \text{if } v_i > A_i^u \\ 0, & \text{if } v_i \leq A_i^u \end{cases} \quad d_{v,i}^l = \begin{cases} \frac{|A_i^l - v_i|}{V_i^b}, & \text{if } v_i < A_i^l \\ 0, & \text{if } v_i \geq A_i^l \end{cases} \quad (2.1)$$

In Eq. (2.1), v_i is the voltage magnitude of the i^{th} bus; V_i^b is the base voltage magnitude; A_i^u and A_i^l are the upper and lower boundary of the voltage alarm limit. The normalized deviation of the alarm limit from the secure limit is defined as follows:

$$g_i^u = \frac{|S_i^u - A_i^u|}{V_i^b}, \quad g_i^l = \frac{|S_i^l - A_i^l|}{V_i^b} \quad (2.2)$$

In Eq. (2.2), S_i^u and S_i^l are the upper and lower boundaries of the voltage security limit. For line flow, only the upper boundary of alarm limit and secure limit is needed. The normalized line flow violation of the alarm limit is defined as follows:

$$d_{p,l} = \frac{||P_l| - A_{p,l}|}{\text{Base MVA}}, \text{ if } |P_l| > A_{p,l} \\ d_{p,l} = 0, \quad \text{if } |P_l| \leq A_{p,l} \quad (2.3)$$

In Eq. (2.3), P_l is the power flow of the l^{th} line; $A_{p,l}$ is the alarm limit of the line flow. The normalized deviation of the alarm limit from the security limit is defined as follows:

$$g_{p,l} = \frac{|S_{p,l} - A_{p,l}|}{\text{Base MVA}} \quad (2.4)$$

In Eq. (2.4), $S_{p,l}$ is the security limit of the line flow. Based on the above definitions, the composite security index (SI) for the system is defined as follows [53]:

$$SI = \left[\sum_i \left(\frac{d_{v,i}^u}{g_{v,i}^u} \right)^{2m} + \sum_i \left(\frac{d_{v,i}^l}{g_{v,i}^l} \right)^{2m} + \sum_l \left(\frac{d_{p,l}}{g_{p,l}} \right)^{2m} \right]^{\frac{1}{2m}} \quad (2.5)$$

Eq. (2.5) is based on the concept of a hyper-ellipse inscribed within the hyper-box for measuring limit violation [56], where m is the exponent used in the hyper ellipse equation. In this study, m is set to be 1. A higher value of SI means that the system is at a higher risk level. For example, if both

voltage magnitudes and line flows are within the alarm limit, which means that the system is operating within the secure region, then $d_{v,i}^u$, $d_{v,i}^l$, and $d_{p,l}$ are all zeroes, which leads to a zero SI; if any voltage magnitude or line flow is out of the alarm limit but still within the security limit, which means that the system can maintain the operation for a short time, then d is smaller than g , which leads to a value of SI that is larger than 0, but mostly below 1; last, if any voltage magnitude or line flow is above the security limit, which means that the system is close to collapse, then d will be larger than g , which will definitely leads to an SI larger than 1.

2.3 Deep CNN-based N-1 Contingency Screening

2.3.1 A brief on deep CNN

Deep CNN is a type of ANN with multiple hidden layers. Deep CNN is known for its strong capability in processing data that has a grid-like topology, e.g., image data. Each image can be represented by a 2-D matrix with pixels filled in. The key of deep CNN lies in that it formulates a hierarchical structure that mimics the visual cortex of humans. According to visual neuroscience, in image recognition, our brain first perceives the color and brightness of the observed object, then the edges, angles, lines, and other local details, followed by the shape, texture and more abstract information, and finally the entire image.

The CNN follows the same logic of the visual cortex. It consists of multiple convolutional layers, each of which contains several convolution kernels. Each convolution kernel scans the entire input to capture the detailed local features. All of the captured features will formulate a feature map for the neural network to identify. As the convolutional layer goes deeper, more high-order, and abstract features will be captured, which preserves the most useful information for image recognition. The principle of deep CNN feature extraction is shown in Figure 2.1:

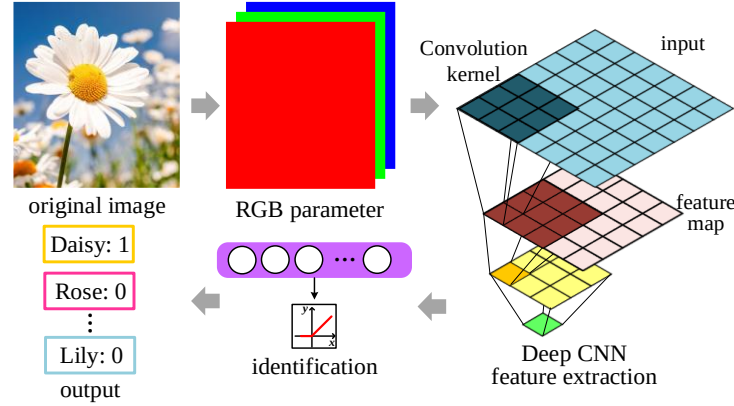


Figure 2.1. The principle of deep CNN feature extraction

Deep CNN has an important feature, which is sparse connectivity [2]. In conventional neural networks, usually every output unit is connected to every input unit. The number of connection parameters that need to be trained can be tremendous. In the case of deep CNN, each output unit in the feature map is only connected to a square patch, named as field of review, from the input that is closest to its location, instead of the entire input. This is called sparse connectivity. The reason of using sparse connectivity is that in one image, one pixel is closely related to its neighboring pixels, but is less related to more distant pixels. Hence, connections between the less related units are removed. With sparse connectivity, the number of parameters for training is greatly reduced, which improves computational efficiency.

2.3.2 Mapping power system data to deep CNN input data

Deep CNN is a natural fit for solving power system problems for two reasons. First, the power system topology has a grid-like structure, and can be fully described by matrices, e.g. nodal admittance matrix, element-bus incidence matrix, branch-path incidence matrix, etc. Second, the power system also possesses the feature of sparse connectivity. The voltage level at one bus is closely related to its neighboring buses, and it is less affected by the buses that are far away. Therefore, a hierarchical deep CNN can learn the element-bus relationship, the line connection, and the entire topology layer by layer based on power system raw data.

In the case of static security assessment, the idea is to apply deep CNN as a classifier for fast system security status classification. The input to the deep CNN will be the power system raw data, including system control variables and system topology, and the output will be the system security status. To realize this function, the first step is to map power system raw data to a grid-like structure for the CNN to read. Following the composition of 2-D image data, an n -bus power system can be represented by four 2-D matrices as shown below:

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nn} \end{bmatrix}_{n \times n} \quad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}_{n \times n} \quad (2.6)$$

$$\mathbf{P} = \begin{bmatrix} p_{inj,1} & 0 & \cdots & 0 \\ 0 & p_{inj,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{inj,n} \end{bmatrix}_{n \times n} \quad \mathbf{Q} = \begin{bmatrix} q_{inj,1} & 0 & \cdots & 0 \\ 0 & q_{inj,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{inj,n} \end{bmatrix}_{n \times n} \quad (2.7)$$

In Eq. (2.6)-(2.7), matrices $\mathbf{G}$ and $\mathbf{B}$ are the bus conductance matrix and bus susceptance matrix, respectively; matrices $\mathbf{P}$ and $\mathbf{Q}$ are the bus active power injection and bus reactive power injection, respectively. Hence, the input data to the deep CNN will have the size of $n \times n \times 4$.

Notice that in Eq. (2.7), the two bus power injection matrices are sparse since only diagonal elements are occupied. To reduce the data size for efficient network training, the bus power injection matrices are further replaced by the following two $1 \times n$ vectors:

$$\mathbf{P} = [p_{inj,1} \quad p_{inj,2} \quad \cdots, \quad p_{inj,n}]_{1 \times n} \quad (2.8)$$

$$\mathbf{Q} = [q_{inj,1} \quad q_{inj,2} \quad \cdots, \quad q_{inj,n}]_{1 \times n}$$

In addition, the bus admittance matrices $\mathbf{G}$ and $\mathbf{B}$ can also be simplified. The aim of inputting the bus admittance matrix is to indicate whether there is a change in system topology during N-1 contingency. In this study, we mainly consider N-1 line outage. Whenever there is a line outage, the self-susceptance element in $\mathbf{B}$ matrix will have a different value, but not necessarily the self-conductance element, since some lines have zero resistance. Hence, we can eliminate the $\mathbf{G}$ matrix

and the non-diagonal elements in $\mathbf{B}$ matrix, and only keep its diagonal elements to represent the system topology, as shown in Eq.(2.9):

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{22} & \cdots & b_{nn} \end{bmatrix}_{1 \times n} \quad (2.9)$$

In this fashion, the original n -bus power system can be equivalently represented by three $1 \times n$ vectors, and the dimension of input data becomes $1 \times n \times 3$. Compared with the original size $n \times n \times 4$, the volume of input data is greatly decreased, which saves both storage space and computation efforts.

2.3.3 Constructing deep convolutional neural network

1) An illustration of convolution operation

In a CNN, the core component is the convolutional layer. A convolutional layer is composed of trainable convolution kernels, or the filter. The function of the filter is to extract features from the input to generate feature maps that are representatives of the input. The feature extraction can be mathematically expressed as follows:

$$I_{new}(i, j) = \sum_{u=0}^{c-1} \sum_{v=0}^{c-1} I(u+i, v+j) \cdot \omega(u, v) + b \quad (2.10)$$

In Eq. (2.10), $I_{new}(i, j)$ is a single unit in the newly generated feature map I from the convolutional layer; $I(u, v)$ is a single unit in the original input; $\omega(u, v)$ is a single unit in the filter, which is also called the weight parameter; c is the size of the filter; b is the bias. Figure 2.2 gives an illustrative example of the above convolution operation:

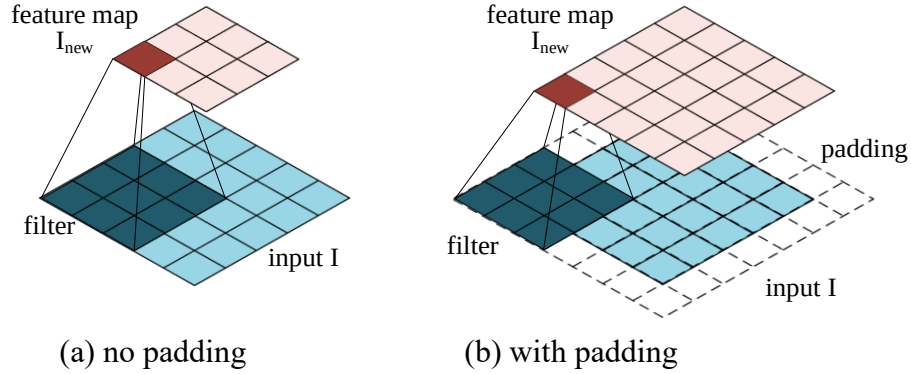


Figure 2.2. Illustration of convolution operation

In Figure 2.2(a), the size of the input is 5×5 , the size of the filter is 3×3 . Each unit in the feature map is the weighted sum of 9 units in the input. The filter scans the input with a step size of 1, hence the size of the feature map is 3×3 . The feature map thus contains the aggregated information from the input. If we want to keep the size of the input, a padding method can be used, as shown in Figure 2.2(b). Two additional rows and columns are added to the input with 0 filled, this is called zero-padding. Then after the convolution operation, the feature map will have the same size as the input.

From the above explanation, it can be observed that the feature extraction function of the convolutional layer refers to adding different weights to the inputs. In conventional machine learning, usually the features of the input have to be computed and selected manually to feed to the neural network for the algorithm to learn. In the case of deep CNN, because of the existence of multiple hidden layers and multiple filters in each layer, the features of the input can be automatically captured by the filters through changing the weights. After the deep CNN is well trained, the weights of the filters will have been properly selected so that the most obvious features from the input will have larger weights, while the less important features are neglected. In this way, the desired output can be obtained. In a convolutional layer, there usually exists multiple filters, and each filter will generate a different feature map. The purpose of utilizing multiple filters is to observe the input from different perspectives, i.e. assigning different weights to the same input unit, so that a comprehensive feature extraction can be obtained. The above explains the automatic feature

extraction ability of the deep CNN.

2) Back propagation algorithm

Before training the neural network, a loss function is defined to describe the accuracy of the deep CNN output. A lower loss indicates higher accuracy of the model. In the N-1 contingency screening problem, we would like deep CNN to realize two goals: as an AC power flow (ACPF) regression tool to generate power system state parameters, i.e., voltage magnitude and voltage angle, and as a classifier for categorizing system security status, which is a multi-task learning model. The loss function for this multi-task learning model is defined as follows:

$$L = \frac{1}{N_s} \sum_{s=1}^{N_s} \left(\frac{1}{n} \left(\sum_{i=1}^n (\theta_{i,s}^* - \theta_{i,s})^2 + \sum_{i=1}^n (v_{i,s}^* - v_{i,s})^2 \right) - y_s^* \log(y_s) \right) \quad (2.11)$$

In Eq. (2.11), N_s is the number of training samples, and n is the number of buses. The first two terms are the mean square error (MSE) of the bus voltage variables, where $\theta_{i,s}$ and $v_{i,s}$ are the deep CNN estimated voltage values, and $\theta_{i,s}^*$ and $v_{i,s}^*$ are the actual voltage values. The third term is called the cross-entropy, where y_s is the deep CNN estimated security classification, and y_s^* is the actual security classification. The cross-entropy is the most widely used loss function for multi-classification problems. In statistics, minimizing cross-entropy is equivalent to maximizing the maximum likelihood. One advantage of applying cross-entropy as loss function is that it barely experiences gradient saturation, which facilitates algorithm convergence

Furthermore, to avoid the issue of overfitting, which is a common problem in regression analysis due to the existence of abnormal values, we add L_2 regularization to the loss function (2.11). Generalization means that the well-trained neural network can be effective across a wide range of inputs, not just the training data that has been fed to the neural network for learning. Sometimes a deep CNN can grow very complex with large values as its weights and biases, where instead of understanding the data, the deep CNN will memorize the one-to-one mapping between the input and the output, which leads to the result that the deep CNN fits well on the training set, but it has poor performance on the test set. This is because all the data in the test set are unseen by deep

CNN, and it has no memorized information for the new samples. The above problem is called overfitting.

L_2 regularization is a common method to overcome the issue of overfitting. L_2 regularization refers to a norm-2 penalty of weight parameters, as shown in Eq. (2.12):

$$L = \frac{1}{N_s} \sum_{s=1}^{N_s} \left(\frac{1}{n} \left(\sum_{i=1}^n (\theta_{i,s}^* - \theta_{i,s})^2 + \sum_{i=1}^n (v_{i,s}^* - v_{i,s})^2 \right) - y_s^* \log(y_s) \right) + \frac{\alpha}{2} \boldsymbol{\omega}^T \boldsymbol{\omega} \quad (2.12)$$

In Eq. (2.12), α is called the regularization parameter, which is a positive number. The penalty term $\boldsymbol{\omega}^T \boldsymbol{\omega} / 2$ stands for model complexity. An overfitted model that intends to match all the input samples, including abnormal values and noises, will have higher model complexity. By adding the penalty term to the loss function, the value of weight parameters will be decreased, and the model will evolve toward low complexity and high generalization.

Upon the definition of loss function, the network weights and biases are updated via a back-propagation algorithm, which is shown as follows:

$$\omega_l^{(k+1)} = \omega_l^{(k)} - \eta \frac{\partial L}{\partial J_{N_L-1}^{(k)}} \cdot \frac{\partial J_{N_L-1}^{(k)}}{\partial J_{N_L-2}^{(k)}} \cdots \frac{\partial J_l^{(k)}}{\partial \omega_l^{(k)}} \quad (2.13)$$

$$b_l^{(k+1)} = b_l^{(k)} - \eta \frac{\partial L}{\partial J_{N_L-1}^{(k)}} \cdot \frac{\partial J_{N_L-1}^{(k)}}{\partial J_{N_L-2}^{(k)}} \cdots \frac{\partial J_l^{(k)}}{\partial b_l^{(k)}} \quad (2.14)$$

In Eq. (2.13)-(2.14), k is the index of iteration; l is the index of convolutional layer; N_L is the total number of convolutional layer; $J_l^{(k)}$ is the output of the l^{th} layer; η is called the learning rate. Since the deep CNN has multiple convolutional layers, the chain rule is applied to calculate the partial derivative of the parameters at each layer. As can be observed, the back-propagation algorithm is essentially a gradient search method. Once the derivative is calculated, the weights and biases can be manipulated to decrease the loss function to its minimum and to obtain the optimal training results.

3) Design of deep CNN structure

The structure of deep CNN for voltage angle calculation is illustrated in Figure 2.3. It consists

of two convolutional (Conv) layers and three fully-connected (FC) layer. The function of the convolutional layers is to extract features from the input power system raw data. Each convolutional layer is composed of a number of learnable convolution kernels, which are shown as purple squares in Figure 2.3.

In the constructed deep CNN, the convolution kernel size for the two layers are [3, 3, 1, 12] and [3, 3, 12, 24]. The first two figures are the height and the width of the convolution kernel, the third figure is the depth of the kernel, and the last figure is the number of kernels. Zero padding is applied here to maintain the width and the height of the input. The generated feature maps further go through an activation function. The activation function will bring nonlinearity to the regression model. This is because the original mathematical relationship between bus power injection and bus voltage is not linear, and cannot be fully represented by the linear convolution operation in (2.10). The limitation of linear transformations will be overcome by the activation function.

In this study, rectified linear unit (ReLU) is applied as the activation function. The ReLU function has the form of $f(x) = \max(x, 0)$, which is a quasi-linear function. This feature allows it to preserve high generalization ability as a linear model, but also avoids the issue of saturation as in the case of other activation functions, such as sigmoid and tanh function.

The output from the second convolutional layer will go through two separate fully-connected layers, FC1 and FC2. This is because in the designed deep CNN, two types of output will be generated. The first output is the system state variables, i.e., the bus voltage angles and bus voltage magnitudes. The second output is the system security status.

In Figure 2.3, the function of the fully-connected layers, FC1, FC2 and FC3, is to transform all the extracted features from the power system raw data into the desired output via matrix multiplication. For FC3, since its output is the classification of the system security status, the *softmax* function is used as the activation function. The *softmax* function has the following expression:

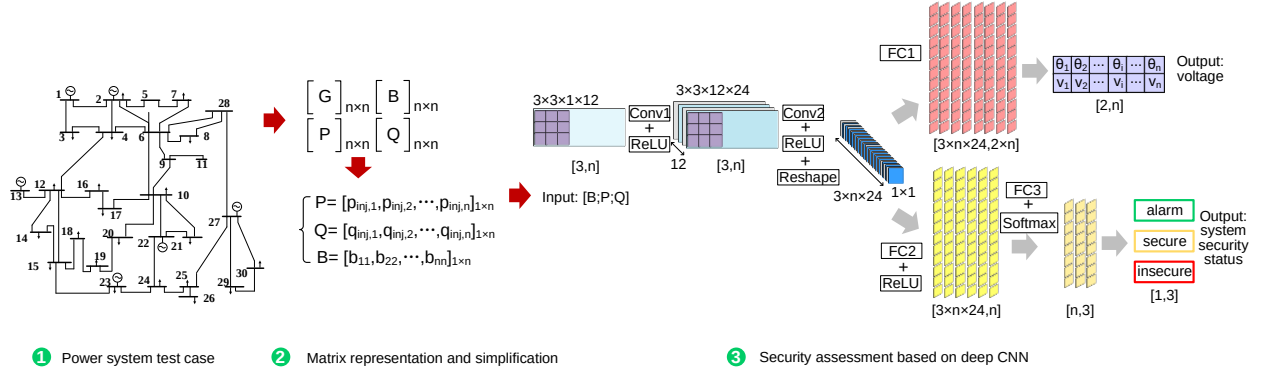


Figure 2.3. Deep CNN structure for N-1 contingency screening

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (2.15)$$

The *softmax* function in Eq.(2.15) normalizes each input element by getting their exponential value divided by the sum of all exponential values. In this way, the difference between any two input elements is enlarged. For example, if $x_i \geq x_j$, then $\exp(x_i)$ will be much larger than $\exp(x_j)$. This higher differentiation among the input can lead to more accurate classification results. The security status with the highest probability is taken as the status for the current system operation.

2.4 Case Study

The proposed image-processing-like, deep CNN model for ACPF calculation under N-1 contingency is tested on the IEEE 9, 30, 57, 118, and 300-bus systems, WECC 181-bus system, and European 1354-bus system to verify its accuracy and computational efficiency. To include multi-scenario uncertainty, Monte Carlo simulation is used to create load and renewable energy variations in training samples. For load uncertainty, we assume that the bus active load follows a uniform distribution within the range of [0.8, 1.2] of the base case, and the bus reactive load is calculated by multiplying the bus active power consumption with a factor uniformly drawn from the range [0.15, 0.25]. For renewable energy uncertainty, we change 40% of the conventional generators in the original test cases into wind generators, and the forecast error of wind generation follows a normal distribution with zero mean and a standard deviation of 0.05. For N-1 contingency,

one line is randomly tripped in each training sample. The hardware environment for deep CNN training is an Nvidia GeForce GTX 1080 Ti Graphic Card with 11 GB memory and 1.582 GHz core clock. The software environment is the open-source deep learning platform TensorFlow for the proposed approach and MATPOWER [57] for the traditional model-based approach. The regression and classification results are shown in Table 2.1.

In Table 2.1, the errors of θ and v are the per unit mean absolute value over the test set compared with the results from model-based ACPF calculation. The classification accuracy is the ratio between the number of test samples that have been correctly classified and the total number of test samples. As the table shows, deep CNN model possesses considerably high accuracy for ACPF calculation, even for large-scale power systems. Also, the training time is within an acceptable range given that the training is completed off-line.

To validate the computational efficiency of deep CNN regression, we compare calculation time of the AC power flow with N-1 contingency using both deep CNN and model-based AC power flow methods, as shown in Table 2.2.

Table 2.1. AC power flow results based on deep CNN

Case	No. of samples		Errors		Training time(s)	Classification Accuracy
	Training	Test	θ	v		
9	3292	1412	6.1e-3	7.2e-4	11.42	97.24%
30	4262	1066	1.5e-3	5.4e-4	23.06	96.25%
57	3360	1440	4.9e-3	1.6e-3	31.59	99.24%
118	3027	1298	7.5e-3	2.9e-4	57.88	100%
181(WECC)	2530	1085	5.7e-2	3.8e-3	65.04	97.70%
300	3445	1477	6.9e-2	2.3e-3	148.91	99.05%
1354 (Eu.)	3981	1707	1.1e-2	1.9e-3	1548.94	96.84%

Table 2.2. Test time comparison

Case	Test size	Test time (s) (deep CNN)	Test time (s) (model-based)	Acceleration ratio
9	1412	0.017	3.500	206
30	1066	0.016	3.303	206
57	1440	0.018	4.323	240
118	1298	0.021	4.905	234
181 (WECC)	1085	0.025	4.655	186
300	1477	0.044	10.15	231
1354 (Eu.)	1707	0.264	34.13	129

In Table 2.2, the last column shows the acceleration based on the computing time of deep CNN and the model-based ACPF method. The deep CNN approach is from 129 to 240 times faster than the latter, with an average of 205 times faster. This is because the well-trained deep CNN has high generalization to unseen test cases, and it can automatically generate AC power flow results and classify system security status under the new given input without any iterative calculation of power flows.

To further demonstrate the superiority of the proposed deep CNN over traditional ANNs, we design an ANN model with only one hidden layer. The size of the hidden layer is $[3 \times n, 3 \times n \times 24]$, which extracts the same number of features as the deep CNN. Hence the two neural networks are comparable. The regression and classification results of ANN are shown in Table 2.3:

Table 2.3. AC power flow results of ANN

Case	No. of samples		Errors		Training time(s)	Test time (s)	Classification Accuracy
	Training	Test	θ	v			
9	3292	1412	2.0e-2	2.3e-3	6.52	0.011	91.64%
30	4262	1066	9.0e-3	2.4e-3	10.29	0.008	87.43%
57	3360	1440	2.7e-2	9.0e-3	14.88	0.009	92.43%
118	3027	1298	2.7e-2	9.0e-4	36.28	0.017	98.54%
181	2530	1085	1.8e-1	1.3e-2	47.31	0.019	75.94%
300	3445	1477	2.0e-1	5.7e-3	139.04	0.036	78.00%
1354	3981	1707	-	-	-	-	-

The results of the 1354-bus system are not available for ANN because the large-scale training data cause memory overflow. For all the other systems, deep CNN provides more accurate classification and regression results than the traditional shallow ANN. This is because the multiple convolutional layers within the deep CNN can extract better features for classification and regression, and this is the key contributing factor to the recent success of CNN in other applications. In addition, the traditional ANN is composed of fully-connected layers, where each neuron is connected to all of the subsequent neurons. This requires more neural parameters and computation. While in deep CNN, the sparse connectivity reduces both redundancy and computation to achieve better accuracy and computational efficiency. Note, although the deep CNN requires longer training time, it is of less importance since training is done offline.

2.5 Conclusions

In this chapter, a data-driven method based on deep CNN is applied for fast N-1 contingency screening. The deep CNN is constructed as both a regression tool and a classifier to evaluate system security status based on power system raw data. With the proposed deep CNN, no power flow calculation is required, which greatly spares computational effort. The simulation results on IEEE test cases verify the classification accuracy of the deep CNN. In addition, comparison with the model-based ACPF substantiates its high computational efficiency in dealing with unseen instances. Therefore, the proposed deep CNN can be a promising tool for online security assessment as well as other related power system researches.

2.6 Acknowledgement

This work was supported in part by the CURENT research center, in part by the ISO New England, and in part by the NSF under Grant ECCS-1809458.

Chapter 3 Fast Cascading Outage Screening based on Deep CNN and Depth-First Search

In this chapter, a data-driven method is proposed for fast cascading outage screening in power systems. The proposed method is a combination of deep convolutional neural network (deep CNN) and depth-first search (DFS) algorithm. First, deep CNN is constructed as a security assessment tool to evaluate system security status based on observable information. With its automatic feature extraction ability and high generalization, a well-trained deep CNN can estimate AC optimal power flow (ACOPF) results for various uncertain operation scenarios, i.e. fluctuated load and system topology change, in a nearly computation-free manner. Second, a scenario tree is built to represent the potential operation scenarios and the associated cascading outages. The DFS algorithm is developed as a fast screening tool to calculate the expected security index value for each cascading outage path along the entire tree, which can be a reference for system operators to take predictive measures against system collapse. The simulation results of applying the proposed deep CNN and the DFS algorithm on standard test cases verify their accuracy and that their computational efficiency is thousands of times faster than the model-based traditional approach, which implies the great potentials of the proposed algorithm for online applications.

3.1 Introduction

3.1.1 Motivation

Protecting the bulk power system against cascading outages is crucial to enhancing the system-wide operation economy and resilience. According to the definition of NERC [58], the cascading outage refers to the situation where the system uncontrollably and successively loses elements triggered by an initial incident at any location. Cascading outage will result in widespread electric service interruption, which cannot be restrained from sequentially spreading beyond an area predetermined by studies. However, the recently growing penetration of uncertainties into the bulk

power system has increased the system vulnerability, as well as the chance for cascading outages. Although the probability for cascading outage to induce blackouts is tiny, the consequences can be catastrophic, resulting in tremendous economic losses and social impacts.

There have been several large-scale blackouts caused by cascading outages in recent years, such as the western U.S. blackout in 1996 [59], the U.S.-Canadian blackout in 2003 [60] and the Arizona-California. blackout in 2011 [61]. Given the costly effects of the cascading outages, NERC has required that each Transmission Planner and Planning Coordinator shall define the criteria or methodology used in the analysis to identify system instability for cascading or uncontrolled islanding during planning assessment studies [62].

Based on the above context, both research communities and the industries have devoted substantial endeavors on cascading outage studies. However, the majority of the existing studies are founded on the conventional model-based method for cascading outage analysis, which suffers from certain computational limitations. Motivated by this consideration, in this chapter we propose a data-driven method that combines the deep CNN with DFS algorithm for a fast cascading outage screening and risk assessment, which aims at potential online applications under uncertain scenarios. More detailed literature review and contributions of this chapter will be presented in the following subsections.

3.1.2 Literature review

The existing research works regarding cascading outage can be mainly classified into three categories: cascading outage simulation and pattern recognition, system vulnerability detection and risk assessment, and post-outage recovery.

In regard to the first category, simulation models have been developed to study the impact of cascading outages, such models include OPA model and its multiple improved versions [63]-[64], Manchester model [65], and CASCADE model [66]. Refs. [67]-[68] further develop multi-timescale cascading outage models to study both slow dynamics like thermal transient and fast dynamics like electrical instability. Ref. [69] proposes a sequential importance sampling strategy

to reduce the number of cascading failures, while still capturing very rare events. To gain better statistic insights into the pattern of the cascading outages propagation, refs. [70]-[71] apply a Markov chain approach, where transition probabilities are estimated from historical data; while refs. [72]-[73] utilize the expectation maximization method, in which the parameters of the probability function are their maximum likelihood estimates.

With respect to system vulnerability detection and risk assessment, a forward-backward Markovian tree search algorithm is introduced in [74], where the risk of the current outage is the expected risk of all its following outages. Based on this work, ref. [75] further considers weather impacts on the line outage probabilities and the system risks, and it develops the associated analytical probability model. Ref. [76] studies the quantitative relationship between the component failure probability and the blackout risks during cascading outages, which can be used as an effective risk assessment tool under the system components change. Ref. [77] shows that the cascading outage risk can be underestimated if not considering the multiple solutions of DC optimal power flow (DCOPF) models, and then proposes remedial measures. Ref. [78] proposes a fast screening method for vulnerable transmission lines based on PageRank algorithm, where a vulnerability degree of each line is calculated based on its post-contingency flow under the N-1 contingency of all the other lines. Ref. [79] defines a branch loading assessment index and designs a cascading fault graph based on the proposed index to demonstrate the vulnerability of each transmission line.

For post-outage recovery measures, simulation-based optimization method [80], multi-agent system method [81], and Markovian tree search method [82] are introduced to reduce the risk mitigation cost through generator re-dispatch and transmission capacity allocation.

The above concern motivates the development of the data-driven method as a meaningful alternative for fast cascading outage screening. As opposite to the model-based method, the data-driven method formulates an approximate mapping between the input and the output. And once the algorithm is well-trained, it is a generalized model that can automatically produce outputs from

unseen new inputs, without a massive amount of analytical computation. Therefore, the data-driven method can be promising for future online cascading outage analysis with real-time data input.

The application of the data-driven method in cascading outage analysis is still at its initial stage in literature. Although some works have been dedicated to utilizing machine learning methods, e.g., ANN [56], CNN [23] and deep autoencoder [24], for security assessment under contingency, few of them has considered the risk of the following cascading outages, which may cause the violation of NERC security standards. In [83], a three-stage decision tree method is proposed to classify the severity level of the cascading blackout. The system states obtained from wide area measurement system (WAMS) are used to train the decision trees, which proves to have a high classification accuracy. In [84], the authors propose a Monte Carlo cascading failure simulation method utilizing the existing model-based software package and a risk assessment method of cascade path based on decorrelated neural network ensembles. However, in this last work, the line flow is used as input to the neural network for system risk evaluation, which implies that the power flow calculation is still needed for new test cases. The ultimate goal of the data-driven method is to utilize the direct system observations, i.e. topologies, as the input to the algorithm without any additional analytical calculation for indirect measurements (such as line flow) to realize a nearly computation-free manner. Otherwise, the data-driven method can still be computation-inefficient for online applications under uncertainties.

3.1.3 Contributions

Based on the previous works, in this chapter, we propose a novel data-driven method for fast cascading outage screening and risk assessment. The proposed method is a combination of deep CNN and DFS algorithm. First, the deep CNN is constructed as a regression tool of the AC optimal power flow (ACOPF) model to quickly obtain the system state variables. The state variables are then utilized to calculate a security index for evaluating outage severity. Secondly, a scenario tree is built to represent all the potential cascading paths in real-time uncertain scenarios. Also, a DFS

algorithm is utilized to screen all the cascading outage paths in the scenario tree to detect the severest path. The detection is based on the estimated security index value from deep CNN. The screening results can serve as a reference for system operators to take corrective measures against system collapse. The main contributions of this chapter are summarized as follows:

1) We propose the deep CNN as an efficient regression method for approximating ACOPF calculation. Unlike other data-driven methods that rely on system state variables as input, a well-trained deep CNN only needs direct observations, e.g., system topology and bus power injection, and will automatically generate the state variables for evaluating outage severity. Hence, it can be directly applied to new test cases without the computationally intensive power flow calculation.

2) We establish a multi-scenario tree as an efficient representation of all the potential cascading outage paths with uncertainties involved. Furthermore, we apply DFS method for a fast cascading outage screening over the entire tree. The DFS method aims to calculate the expected accumulative security index of each cascading outage path for evaluating their severity. With a proper screening order of all the cascading outages, the proposed DFS can complete the traversal with extremely low elapsed time, which is highly applicable in the case of large-scale power system cascading outage screening.

The rest of this chapter is organized as follows: section 3.2 demonstrates the design of the proposed deep CNN for ACOPF regression; section 3.3 explains the construction of the scenario tree and the details of the DFS algorithm; section 3.4 verifies the proposed deep CNN and DFS algorithm for cascading outage screening on standard test cases; finally, section 3.5 concludes the chapter.

3.2 Deep CNN-based Security Assessment

3.2.1 Mapping power grid data to deep CNN input data

In the case of system security assessment, the function of deep CNN is to approximate ACOPF calculation and to obtain the system state variables. The state variables can then be used to calculate the security index to evaluate system security status. To achieve this function, the first step is to

map power system raw data to a grid-like structure for the CNN to analyze.

The ACOPF model is shown as follows:

$$\min \sum_{g=1}^{N_g} C_p(P_g) + C_Q(Q_g) \quad (3.1)$$

$$\text{s.t. } \sum_{g \in i} P_g - \sum_{d \in i} P_d = v_i \sum_{j \in i} v_j (g_{ij} \cos \theta_{ij} + b_{ij} \sin \theta_{ij}) \quad (3.2)$$

$$\text{s.t. } \sum_{g \in i} Q_g - \sum_{d \in i} Q_d = v_i \sum_{j \in i} v_j (g_{ij} \sin \theta_{ij} - b_{ij} \cos \theta_{ij}) \quad (3.3)$$

$$-F_{ij,\max} \leq |F_{ij}| \leq F_{ij,\max} \quad (3.4)$$

$$v_{i,\min} \leq v_i \leq v_{i,\max} \quad (3.5)$$

$$P_{g,\min} \leq P_g \leq P_{g,\max}, \quad Q_{g,\min} \leq Q_g \leq Q_{g,\max} \quad (3.6)$$

In Eq. (3.1)-(3.6), the known parameters are the bus active/reactive load P_d , Q_d , and system topology g_{ij} , b_{ij} , which will be the input to the deep CNN; and the unknowns are the bus active/reactive generation, P_g , Q_g , bus voltage magnitude v_i , and bus voltage angle θ_i . Given that we only need bus voltage for security index calculation, the deep CNN will only output v_i and θ_i in this case. However, notice that the input parameters differ in their dimension. Given an n -bus power system, the P_d and Q_d will both be $1 \times n$ vectors, while g_{ij} and b_{ij} are both $n \times n$ matrices. Deep CNN requires that the input known quantities should have the same dimensions. For example, for the image data, each image has the following dimensions: w (width) $\times h$ (height) $\times c$ (number of color channels), where the dimensions for each color channel are the same. To reach this requirement, we utilize the following $1 \times n$ vector to represent system topology:

$$\text{diag}(\text{imag}(Y)) = \text{diag}(B) = [b_{11} \quad b_{22} \quad \dots \quad b_{nn}] \quad (3.7)$$

In Eq. (3.7), Y is the bus admittance matrix, and B is the bus susceptance matrix. The reason for utilizing the bus self-susceptance elements to represent system topology change is that whenever there is a line outage, the self-susceptance elements will definitely change, but not necessarily the self-conductance elements, since some lines have zero resistance. By removing the non-diagonal

elements in the B matrix, we only keep the most dominant elements as an efficient representation of system topology. Since deep CNN regression is a data-driven method, the regression error caused by the missing data in the G matrix and the B matrix will be automatically made up via iterative training based on existing data samples. With the above simplification, a deep-CNN regression for ACOPF calculation only requires three $1 \times n$ vectors as the input. The volume of training data is acceptable even in case of large-scale power systems.

In the above static security assessment problem, mean square error (MSE) is used as the loss function:

$$L = \frac{1}{N_S} \sum_{s=1}^{N_S} \left(\frac{1}{n} \sum_{i=1}^n (\theta_{i,s}^* - \theta_{i,s})^2 + \frac{1}{n} \sum_{i=1}^n (v_{i,s}^* - v_{i,s})^2 + (SI_s^* - SI_s)^2 \right) + \frac{\alpha}{2} \boldsymbol{\omega}^T \boldsymbol{\omega} \quad (3.8)$$

In Eq. (3.8), N_S is the number of training samples, n is the number of power system buses, $\theta_{i,s}^*$ and $v_{i,s}^*$ are the desired output from the deep CNN, i.e., the actual bus voltage angle and bus voltage magnitude, $\theta_{i,s}$ and $v_{i,s}$ are the estimated bus voltage angle and bus voltage magnitude. Since we need to evaluate the system security status, a third term is added to the loss function, which is the difference between the actual security index value SI_s^* and the estimated security index value SI_s . The objective of deep CNN is to minimize the deviation between the estimation and the ground truth to formulate an accurate enough ACOPF regression model. The last item in Eq. (3.8) is the L_2 regularization, which is to avoid the issue of overfitting.

3.2.2 Constructing deep convolutional neural network

The structure of the deep CNN is demonstrated in Figure 3.1. It consists of two convolutional (Conv) layers and five fully-connected (FC) layers. The functions of these deep CNN layers are explained in detail as follows:

a) The input data is a $3 \times n$ matrix, where n is the number of buses. These $3 \times n$ data corresponds to three $1 \times n$ vectors, i.e., the real loads of n buses, the reactive loads of n buses, and the n diagonal elements of the B matrix.

The first convolutional layer has a filter with the size of $[3, 3, 1, 12]$, where the first three numbers

are the height, width, and the depth of the filter. The last figure is the number of filters. In this layer, 12 filters will be sampling the input data. As a result, the input data is deepened after scanned by the filter. In addition, the zero-padding is applied to maintain the original size of the input data. Hence the output of the first convolutional layer has the size of $[3, n, 12]$.

The filter has the size of 3×3 , which means that it assumes the three neighboring buses have strong interrelations, e.g., bus 2-4, bus 3-5, since each time the filter samples a size of 3×3 from the input. This is in accordance with physical laws because the bus voltage angle is most affected by its closest neighboring buses. The size of the filter can also be increased to include more neighboring buses, but this comes with a larger quantity of parameters that need to be trained.

b) The output from the first convolutional layer will go through an activation function. The activation function will add nonlinearity to the feature extraction. This is because Eq. (2.10) is a linear transformation. However, the ACOPF model (3.1)-(3.6) is nonlinear and nonconvex. Introducing the activation function to feature extraction can remove the limitation of linear representation. The ReLU is used as the activation function.

c) The output from the ReLU function will go through the next convolutional layer, which has filters with the size of $[3, 3, 12, 24]$. More features are extracted by the second convolutional layer.

d) The output from the second convolutional layer has the size of $[3, n, 24]$, which is a 3-D tensor. It is further flattened as a $1 \times (3 \times n \times 24)$ vector and goes through a fully-connected layer. In the fully-connected layer, there is a connection between each neuron and each element in the input. In this case, the size of the weight parameters in the fully-connected layer is $[3 \times n \times 24, 2 \times n]$, and the size of the bias is $2 \times n$. So that after the matrix multiplication, the output will become a vector with the size of $[1, 2 \times n]$, which is a combination of n bus voltage magnitudes and n bus voltage angles.

e) Because we need to evaluate the system security status, the obtained voltage variables will further go through the next four FC layers to calculate the security index, which is a regression of Eq. (2.5). Before sending the voltage variables to the FC layer, the diagonal elements of B matrix are added to the voltage tensor to reflect the system topology change. This is because in Eq. (2.5),

the line flow is related to system topology.

The four following FC layers have the size of $[3 \times n, 12 \times n]$, $[12 \times n, 6 \times n]$, $[6 \times n, n]$, and $[n, 1]$, respectively. After the matrix multiplication, the final output will be a 1×1 scalar, which is the security index value.

Via the above deep CNN, both the system state variables and system security index can be obtained. Some may argue that since we only need the security index to evaluate system status, there is no need to output the bus voltage variable, which may result in a less complicated neural network structure. However, the security index value only shows the system security status as a whole, and it cannot reflect the local weakness and vulnerability. With system state variables, we can gain insights into the local voltage margin and line flow margin. In summary, the state variables cover more detailed information of system operation than the security index value.

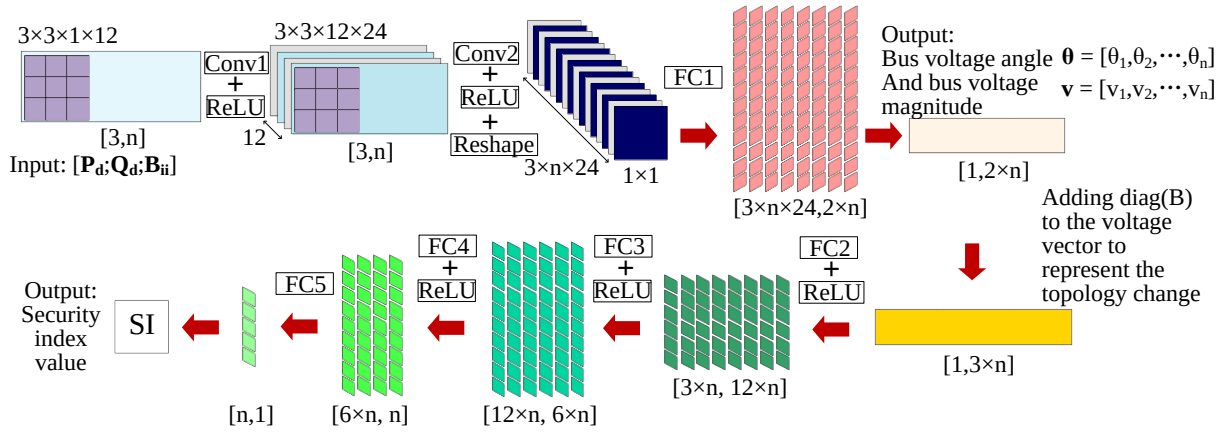


Figure 3.1. Deep CNN structure for security assessment

3.2.3 Training sample generation

In the training phase of the deep CNN, large quantities of training samples are required for fine-tuning the neural network parameter. Since the proposed deep CNN is designed for cascading outage analysis, in the training sample, power flow results for k outage stages are included, where k indicates the number of electrical components that are out of service. In this study, we mainly consider line outage contingency. During power system operation, once a transmission line is tripped, it may cause overloading of other transmission lines and induces cascading line outages. The probability of the l^{th} transmission line failure is calculated as follows [84]:

$$p_l = \begin{cases} 1, & |P_l| \geq S_{p,l} \\ \frac{|P_l| - A_{p,l}}{S_{p,l} - A_{p,l}}, & A_{p,l} \leq |P_l| \leq S_{p,l} \end{cases} \quad (3.9)$$

At each outage stage, based on Eq. (3.9), the line with the highest failure probability is selected as the tripped line. The whole process of generating cascading contingency training samples is shown in Figure 3.2, and is explained as follows:

- 1) To begin with, an operation scenario is randomly generated based on Monte-Carlo simulation to represent real-time uncertainties. In this study, we mainly consider the load variations;
- 2) Under the generated scenario, the model-based ACOPF is conducted to evaluate system security status; the system parameters and power flow results are stored for future training of deep CNN;
- 3) Based on the obtained power flow results, the tripped line is selected according to Eq. (3.9). If there are several lines that are out of limit, the line with the highest probability is selected as the tripped line;
- 4) Since we consider cascading outages in this study, if the number of line outage stages reaches k , then go to step 5); else go back to steps 2)-3) to repeat the above process;
- 5) If enough operation scenarios have been generated, then the whole process is complete; else go back to step 1) to regenerate operation scenarios and repeat the above cascading outage process.

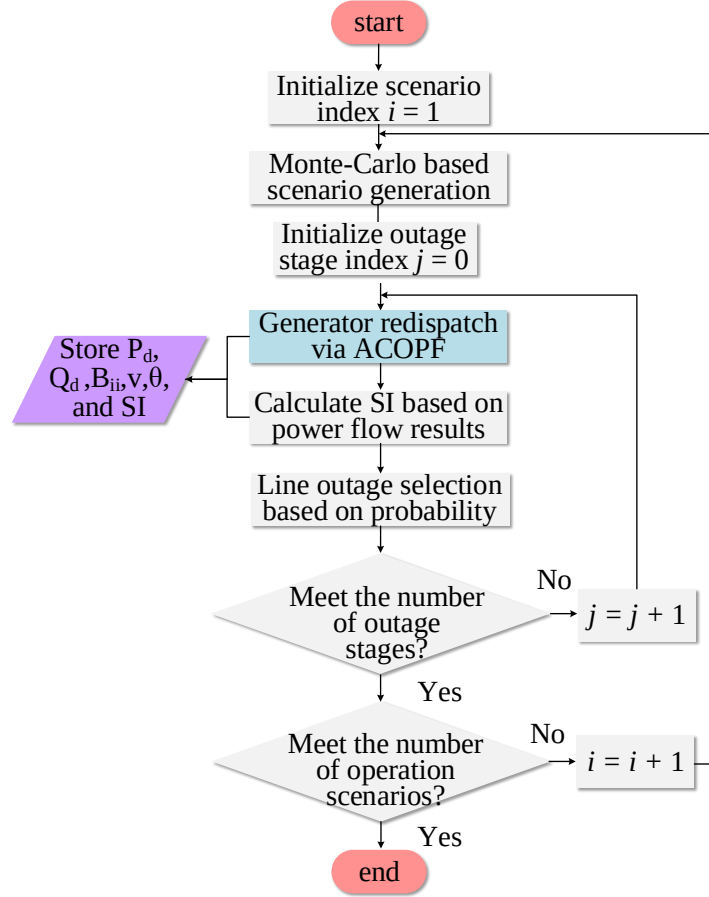


Figure 3.2. Flowchart of generating cascading outage training samples

3.3 Cascading Outage Screening based on Depth-First Search Algorithm

In the previous section, deep CNN is constructed to approximate ACOPF for evaluating system security status. In this section, we will demonstrate how to apply the calculated security index in cascading outage screening under multiple real-time scenarios.

Given that the cascading outage is a time sequential process, we construct a scenario tree to represent the continuous dynamic changes of the system operation scenarios, which is shown in Figure 3.3 [47].

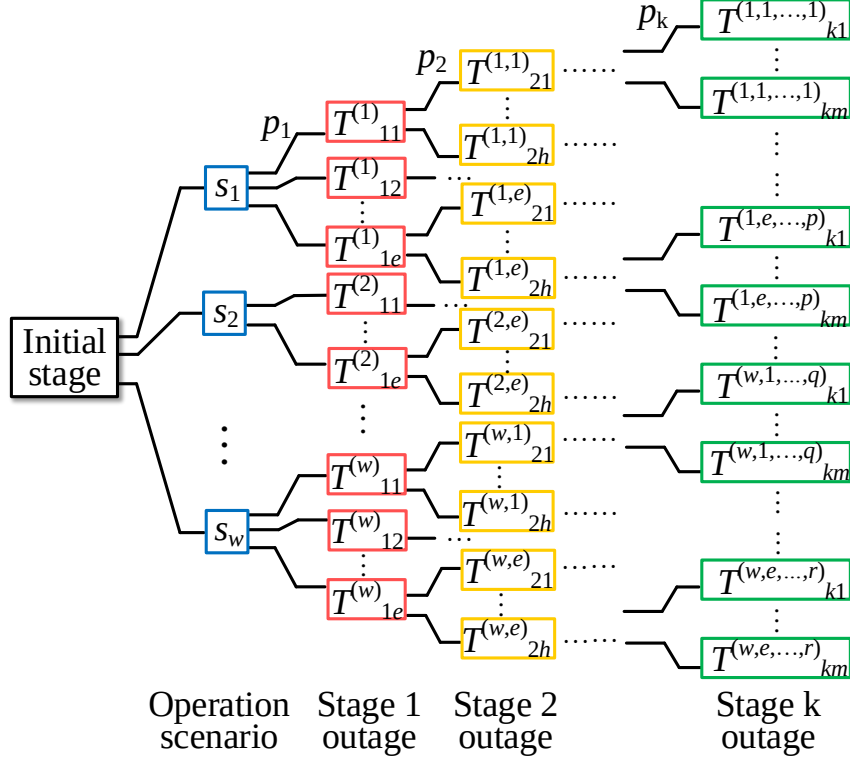


Figure 3.3. Multi-scenario tree for cascading outage screening

Figure 3.3 corresponds with the process of training sample generation shown in Figure 3.2. In Figure 3.3, beginning at the initial stage, different operation scenarios are first generated to represent the real-time uncertainties using Monte Carlo simulation. The uncertainties are regarded as a disturbance to trigger the following cascading line outages. At each tree node, i.e., at each outage stage, T stands for system topology, the superscript records all the previous stages, and the subscript indicates the current stage. Take $T_{21}^{(1,1)}$ as an example, in the superscript “(1,1)”, the first “1” indicates the operation scenario 1, and the second “1” indicates the first line outage scenario in the 1st outage stage; in the subscript “21”, the “2” indicates the 2nd outage stage, and the “1” indicates the first line outage scenario in the 2nd outage stage.

On each branch that connects two tree nodes, p_k is the line failure probability, which can be calculated by Eq. (3.9). A cascading outage path is defined as a path that starts from the initial stage and terminates at the k^{th} outage stage. A value is assigned to each node along the path, namely

the security index SI . The goal of cascading outage screening is to evaluate the severity of each cascading outage path based on SI .

We define the following accumulative security index for severity measurement:

$$\begin{aligned} SI_{\text{exp}}^{(k)} &= p_k SI^{(k)} \\ SI_{\text{exp}}^{(i-1)} &= p_{i-1} (SI^{(i-1)} + SI_{\text{exp}}^{(i)}), \text{ for } i = k, k-2, \dots, 2 \end{aligned} \quad (3.10)$$

In Eq. (3.10), starting from the k^{th} outage stage, the accumulative security index is calculated in a recursive manner. For example, for the cascading outage path $s_1 \rightarrow T_{11}^{(1)} \rightarrow T_{21}^{(1,1)} \rightarrow \dots \rightarrow T_{k1}^{(1,1,\dots,1)}$, the accumulative security index is calculated as follows:

$$\begin{aligned} T_{k1}^{(1,1,\dots,1)} : SI_{\text{exp}}^{(k)} &= p_k SI^{(k)} \\ T_{(k-1)1}^{(1,\dots,1)} : SI_{\text{exp}}^{(k-1)} &= p_{k-1} (SI^{(k-1)} + SI_{\text{exp}}^{(k)}) \\ &\dots\dots\dots \\ T_{21}^{(1,1)} : SI_{\text{exp}}^{(2)} &= p_2 (SI^{(2)} + SI_{\text{exp}}^{(3)}) \\ T_{11}^{(1)} : SI_{\text{exp}}^{(1)} &= p_1 (SI^{(1)} + SI_{\text{exp}}^{(2)}) \end{aligned} \quad (3.11)$$

Finally, $SI_{\text{exp}}^{(1)}$ is taken as the final accumulative value of the entire cascading outage path.

Based on Eq.(3.10), we design the following DFS algorithm for calculating the accumulative security index for each cascading outage path, as shown in Figure 3.4. The main idea of the DFS algorithm is to first explore the cascading outage stages along one path as deep as possible until reaching the last outage stage, while storing the order of line outages and the associated security index; then backtrack to the previous outage stages and update their expected security index. If all line outages at one outage stage have been scanned, then go back to the previous outage stage and switch to another line outage as the source node and repeat the above process, until all the cascading outage paths are screened. The DFS algorithm is a natural fit for the cascading outage screening because its forward-backward propagation corresponds with the recursive calculation of $SI_{\text{exp}}^{(k)}$ in Eq.(3.10).

Note that in the above process, the original security index at each outage stage has already been calculated by deep CNN. Once the deep CNN is well-trained, it can be directly applied to new test cases in the multi-scenario tree and automatically generates ACOPF results and the associated security index, which greatly reduces computational burden. In the next section, the simulation

studies prove that the combination of the deep CNN and the above DFS algorithm makes it possible to scan a large-scale multi-scenario tree with extremely low time cost, while maintaining the desired accuracy.

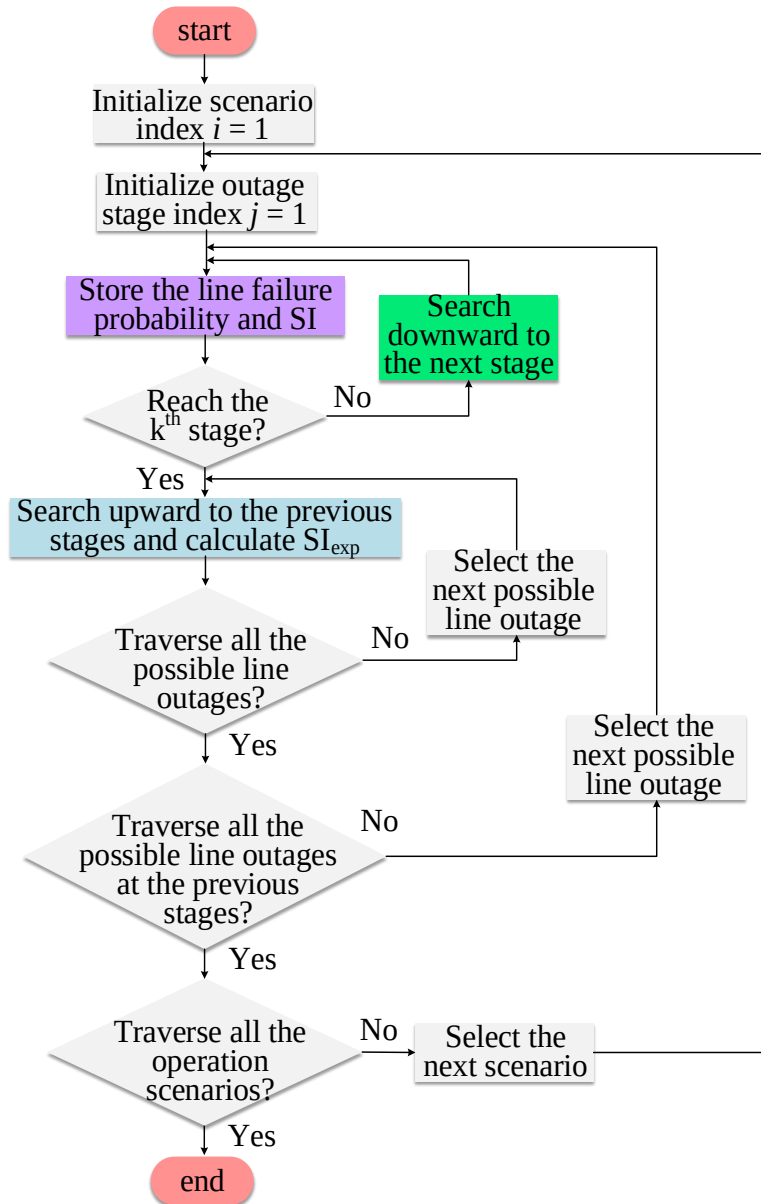


Figure 3.4. Flow chart of depth-first search algorithm

3.4 Simulation Analysis

In this section, we test the proposed deep CNN and DFS method for cascading outage screening on IEEE 57-bus system and European 1354-bus system. Deep CNN is first implemented as a regression model of ACOPF. Then, the scenario tree and DFS algorithm are deployed for fast cascading outage path screening.

3.4.1 Deep CNN regression of ACOPF model

The structure of the proposed deep CNN has been demonstrated in Figure 3.1. For scenario uncertainties, we assume that the variation of load forecast error follows a normal distribution with zero mean and a standard deviation of 0.1. In this study, we consider at most three cascading outage stages, i.e. $k = 3$. The number of operation scenarios and possible line outage scenarios considered in generating the training set and test set are summarized in Table 3.1:

Table 3.1 is explained as follows: taking the IEEE 57-bus system as an example, for the training set, we have 33 different load scenarios at the initial stage. At each outage stage, 10 possible line outage selections are considered based on their failure probability. As such, the total number of training samples will be $N_s + N_s \times N + N_s \times N^2 + \dots + N_s \times N^k = N_s \times (N^{(k+1)} - 1)/(N - 1) = 33 \times (10^4 - 1)/(10 - 1) = 36,663$, where N_s is the number of load scenarios, in this case it is 33; and N is the number of possible line outages, in this case it is 10. However, under some circumstances, the ACOPF does not converge. Such samples are removed from the above samples. The same explanation applies for other figures in the table.

Note that part of the training set is used as the validation set. For both systems, 20% of the training samples are used as the validation set. The difference between the validation set and the test set is that the validation set has load scenarios that are also included in the training set, while the test set has different load scenarios from those in the training set (but follows the same probability distribution). The deep CNN accuracy is verified by both sets to prove its generalization under different instances.

Table 3.1. Summary of training/test set generation

Test case	Training set				Test set			
	No. of scenarios	Stage 1	Stage 2	Stage 3	No. of scenario	Stage 1	Stage 2	Stage 3
57-bus	33	10	10	10	7	10	10	10
1354-bus	61	20	20	20	14	20	20	20

All the samples are generated by the MATLAB toolbox MATPOWER [57]. The hardware environment is an Nvidia GeForce GTX 1080 Ti Graphic Card with 11 GB memory and 1.582 GHz core clock. The software environment is the open-source deep learning platform TensorFlow. The learning rate is set to 1e-3, and the number of training epochs is set to 500. To improve the deep CNN regression accuracy, a repeated training process is conducted. For example, with the 57-bus system, the training process for deep CNN is repeated for 3 times. Each time the learning rate is scaled down 10 times from its previous value. This means that the deep CNN is first trained for 500 epochs with the learning rate 1e-3, and the trained model is saved. Then the deep CNN is trained for another 500 epochs with the saved model as the initial value and a learning rate of 1e-4. And the new trained model is saved. The above process repeats for 3 times. For the 1354-bus system, the process repeats for 4 times. With the repeated training, the algorithm can fine search within the local area with a smaller learning rate to avoid the overshooting. The final training results and the test results are shown in Table 3.2-Table 3.3:

Table 3.2. Sample set size for deep CNN training and testing

Case	Training set size	Validation set size	Test set size
57-bus	24,620	6,155	5,497
1354-bus	18,680	4,670	5,278

Table 3.3. ACOPF regression results based on deep CNN

Case	Validation set error			Test set error		
	v	θ	SI(%)	v	θ	SI(%)
57-bus	5.3e-4	9.5e-4	0.65	6.2e-4	1.8e-3	2.80
1354-bus	2.8e-4	2.6e-4	0.09	1.6e-4	2.4e-4	0.15

In Table 3.3, the error of v and θ is the mean absolute difference between the actual value and the estimated value produced by deep CNN, and the error of SI is the mean relative percentage error. As shown in the table, the error measurement is considerably small for both systems, which demonstrates the accuracy of deep CNN regression.

To illustrate the high computational efficiency of the deep CNN, we compare ACOPF runtime of the 5,497 and 5,278 test samples between deep CNN regression and the model-based method in MATPOWER, and the results are summarized in Table 3.4:

As shown in Table 3.4, the computation speed of deep CNN is thousands of times faster than that of the traditional model-based ACOPF. This is because once the deep CNN is well-trained, it has formulated a high dimensional mapping between the input and the output, and it can directly generate optimal power flow results for new instances with different loading conditions and system topology changes, without incurring the iterative calculation. This computation-free feature makes deep CNN an advantageous tool for solving highly complex large-scale power system planning and operation problems, where the model-based method can be excessively time-consuming and resource-consuming. In addition, the training time for both test cases is within an acceptable range, given that the training for deep CNN is completed off-line.

Table 3.4. Test time comparison

Case	Training time(s)	Test time (s) (deep CNN)	Test time (s) (model-based)	Acceleration ratio
57-bus	906	0.16	225.85	1,412
1354-bus	24,692	5.7	8,185	1,436

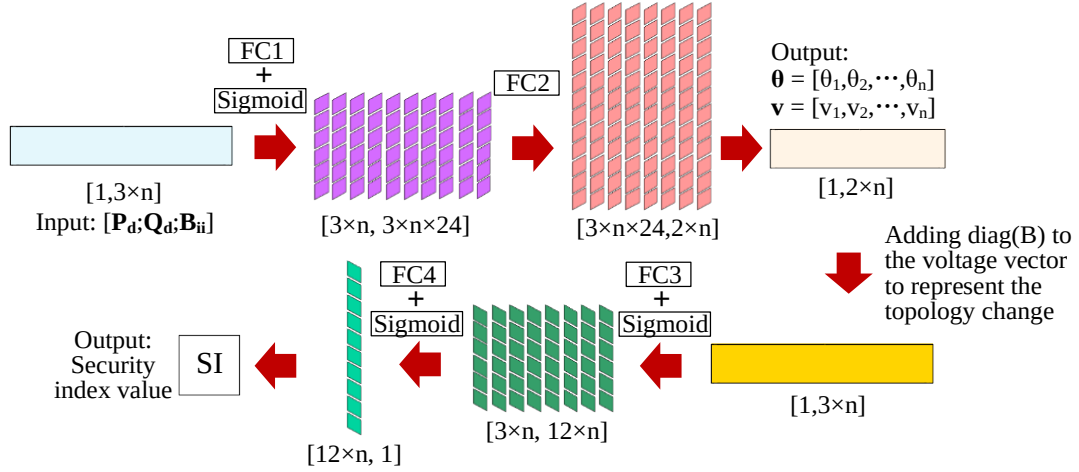


Figure 3.5. ANN structure for security assessment

To further validate the high learning ability of the proposed deep CNN, we design a traditional ANN with fully-connected layers as comparison for cascading outage screening. The configuration of the proposed ANN is shown in Figure 3.5.

The difference between the proposed deep CNN model and the traditional ANN is that the former utilizes the convolutional layers to extract features, while the latter utilizes the fully connected layers. In addition, the deep CNN has multiple hidden layers for sufficient feature extraction, while in ANN, there is only one hidden layer between the input and the output, e.g., FC1 and FC3 are the hidden layers in Fig. 6. The same training set, validation set, and test set are used for ANN training and testing. The ANN is also trained repeatedly for the same number of epochs for fair comparison. The final regression results of ANN are shown in Table 3.5:

Table 3.5. ACOPF regression based on traditional ANN

Case	Validation set error			Test set error		
	ν	θ	SI(%)	ν	θ	SI(%)
57-bus	8.3e-4	1.9e-3	4.62	1.0e-3	3.0e-3	8.78
1354-bus (Eu.)	-	-	-	-	-	-

The results of the 1354-bus system are not available for ANN because the large-scale system causes the size of the FC layer parameters exceeds the memory limit. For the 57-bus system, it can be seen that deep CNN provides more accurate results than the traditional shallow ANN. This is because the convolutional kernels within the deep CNN utilizes the sparse connectivity to extract better features for model regression. In addition, the number of parameters in the convolutional layers is much lower than that of the FC layers, which spares both computation source and storage source.

3.4.2 Identifying cascading outage path with DFS algorithm

The function of deep CNN is to evaluate system security status for each operation scenario during cascading outages. In this subsection, a scenario tree is first constructed to represent the multiple realizations of real-time uncertainties. Then the security index of each node in the scenario tree is calculated based on the estimated results from deep CNN. Finally, the DFS algorithm is applied to evaluate the severity of each cascading outage path along the entire scenario tree.

Two scenario trees for the IEEE-57 bus system and European 1354-bus system are constructed based on their respective test set. For the 57-bus system, because no line capacity data is given, the alarm limit is set as 1.35 times of the line flow under normal conditions, and the security limit is set as 1.4 times of the line capacity, which follows [84]; for 1354-bus system, the alarm limit is set as 1.35 times of the original line capacity, and the security limit is set as 1.4 times of the line capacity. The results of cascading outage screening are shown in Table 3.6-Table 3.8.

Table 3.6. Time efficiency of DFS algorithm

Case	No. of paths	Time(s)	Average $SI_{exp}^{(1)}$ error (%)
57-bus	4,856	0.019	1.06
1354-bus	4,424	0.010	0.16

In Table 3.6, the fourth column is the average relative error of the accumulative security index $SI_{exp}^{(1)}$ based on the estimated results from deep CNN compared with the actual ACOPF results for all the cascading outage paths. It can be seen that the average errors for the two test cases are considerably small, which further indicates that the deep CNN regression results can be utilized as a reliable index for cascading outage severity evaluation.

The DFS algorithm is written in MATLAB R2017b, and the hardware environment is an Nvidia GeForce GTX 1080 Ti Graphic Card with 11 GB memory and 1.582 GHz core clock. As shown in the third column of Table 3.7, the calculation time of $SI_{exp}^{(1)}$ for all the cascading outage paths in both test cases takes no more than 0.02 second, which demonstrates the high computational efficiency of the DFS algorithm.

Table 3.7 presents the cascading outage path with the highest $SI_{exp}^{(1)}$ in the two test cases, which indicates their highest severity. “Actual” means the result is based on the real SI value for calculating $SI_{exp}^{(1)}$, and “Estimated” means that the result is based on the value from deep CNN for calculating $SI_{exp}^{(1)}$. In the third column, e_{load} stands for the load forecast error. For example, in the 57-bus system, the severest cascading outage path is the 7th scenario with a load forecast error of 0.0725, with line 9-11 tripped at the 1st outage stage, line 9-13 tripped at the 2nd outage stage, and line 3-15 tripped at the 3rd outage stage. As shown in the table, in the 57-bus system, the actual cascading outage path is the same as the estimated cascading outage path; in the 1354-bus system, the estimated path is different from the actual path in the third outage stage. However, the estimated path has the third highest $SI_{exp}^{(1)}$ if using the actual SI for calculating, which is only 0.0025% smaller than the highest $SI_{exp}^{(1)}$. Therefore, it can be safely concluded that the computation-free deep CNN is

accurate enough to serve as a highly efficient tool for fast cascading outage screening in combination with the DFS algorithm, especially in the case of large-scale power systems with multiple uncertain scenarios.

Some further insights can be gained from the cascading outage screening results. In Table 3.8, we analyze the transmission lines that are most frequently tripped at each cascading outage stage in the first 100 cascading outage paths with the highest estimated $SI_{exp}^{(1)}$ value, and also compare with the results based on actual $SI_{exp}^{(1)}$ value. The line indices marked in blue are the lines that are missed in the estimated line set. As shown in the table, almost all of the lines in the actual line set are detected in the estimated line set, which again proves the accuracy of deep CNN regression. The information revealed in the table can be used as a reference for system operators to take predictive measures for line capacity expansion or load shedding in advance to improve system operation security against cascading risks.

Table 3.7. Cascading outage screening results

Case		Scenario	Stage 1	Stage 2	Stage 3
57-bus	Actual	7 (e_{load} : 0.0725)	L 9-11	L 9-13	L 3-15
	Estimated	7 (e_{load} : 0.0725)	L 9-11	L 9-13	L 3-15
1354-bus	Actual	13 (e_{load} : -0.0865)	L 2426-8961	L 1146-7945	L 6806-1609
	Estimated	13 (e_{load} : -0.0865)	L 2426-8961	L 1146-7945	L 3248-7309

Table 3.8. Index of most frequently tripped lines in each cascading outage stage

Case 57		
Stage 1	Actual lines	L 9-12, L 12-13, L 11-13, L 9-13, L 4-6, L 9-11
	Estimated lines	L 9-12, L 12-13, L 11-13, L 9-13, L 4-6, L 9-11
Stage 2	Actual lines	L 5-6, L 3-15, L 9-10, L 12-13, L 9-12, L 11-13, L 4-6, L 9-11, L 9-13
	Estimated lines	L 5-6, L 3-15, L 9-10, L 12-13, L 9-12, L 11-13, L 4-6, L 9-11, L 9-13
Stage 3	Actual lines	L 24-26, L 26-27, L 19-20, L 9-11, L 12-13, L 11-13, L 48-49, L 13-14, L 23-24, L 4-6, L 3-15, L 9-13, L 9-10, L 5-6, L 9-12
	Estimated lines	L 24-26, L 26-27, L 19-20, L 9-11, L 12-13, L 11-13, L 48-49, L 13-14, L 23-24, L 4-6, L 3-15, L 9-13, L 9-10, L 5-6, L 9-12
Case 1354		
Stage 1	Actual lines	L 3248-7309, L 4689-4936, L 6629-7309, L 1146-7945, L 2426-8961
	Estimated lines	L 3248-7309, L 4689-4936, L 6629-7309, L 1146-7945, L 2426-8961
Stage 2	Actual lines	L 2426-6888, L 6629-7309, L 3248 – 7309, L 4689-4936, L 1146-7945, L 2426-8961
	Estimated lines	L 2426-6888, L 6629-7309, L 3248 – 7309, L 4689-4936, L 1146-7945, L 2426-8961
Stage 3	Actual lines	L 2426-6888, L 1146-7945, L 6629-7309, L 3248-7309, L 4689-4936, L 2426-8961, L 6806-1609
	Estimated lines	L 2426-6888, L 1146-7945, L 6629-7309, L 3248-7309, L 4689-4936, L 2426-8961, L 6806-1609

3.5 Conclusions

In this chapter, a data-driven fast cascading outage screening approach is proposed based on deep CNN and DFS algorithm. The deep CNN is constructed as a regression tool to estimate the ACOPF results under different contingencies and also the system security index. The DFS

algorithm is applied to scan the scenario tree to detect the most severe cascading outage path based on the estimated security index value provided by deep CNN. Simulation results on the IEEE 57-bus and European 1354-bus systems verify the high accuracy and high computational efficiency of the proposed method. The practical implications of the study are summarized as follows:

1) With the increasing penetrations of uncertainties into the bulk power system, the number of operation scenarios needed to be examined for system security assessment will grow exponentially, which will result in an unbearable computational cost to the conventional model-based methods. The proposed data-driven method with its nearly computation-free fashion can quickly detect the system vulnerability under multiple scenarios. The high accuracy and computational efficiency make the proposed method a desirable choice for real-time system screening.

2) With the historical cascading outage data provided as the training set, the proposed data-driven method can be easily adapted to power systems with different scales and multiple outage stages. The flexibility and scalability give the proposed method the potential to be developed as a general cascading outage screening tool in real-world applications.

3) The screening results of the deep CNN and the DFS method can serve as a reference for power system operators to take preventive measures against the latent outages, and to reduce the system risk management cost such as load shedding and generator redispatch. The screening results can also be used as guidelines for future power system planning to efficiently allocate the investment to the most vulnerable transmission devices.

3.6 Acknowledgement

This work was supported in part by the CURENT research center, ISO New England, and NSF under Grant ECCS-1809458.

Chapter 4 Multi-microgrid Energy Management with Deep Learning and Reinforcement Learning

In this chapter, an intelligent multi-microgrid (MMG) energy management method will be proposed based on deep neural network (DNN) and model-free reinforcement learning (RL) techniques. In the studied problem, multiple microgrids are connected to a main distribution system and they purchase power from the distribution system to maintain local consumption. From the perspective of the distribution system operator (DSO), the target is to decrease the demand-side peak-to-average ratio (PAR), and to maximize the profit from selling energy. To protect user privacy, DSO learns the MMG response by implementing a DNN without direct access to user's information. Further, the DSO selects its retail pricing strategy via a Monte Carlo method from RL, which optimizes the decision based on prediction. The simulation results from the proposed data-driven deep learning method, as well as comparisons with conventional model-based methods, substantiate the effectiveness of the proposed approach in solving power system problems with partial or uncertain information.

4.1 Introduction

The latest advancement of deep learning has opened the door of new AI-driven approaches to solve a broad range of power system problems [85]. Demand-side resource management is one of such problems. In recent years, emerging demand-side resources are playing an increasingly important role in maintaining the economy and security of bulk power system operation [86]-[88]. Many existing research works have been dedicated to exploring the function of multifarious demand-side resources, e.g., distributed generators, plug-in electric vehicles, demand response programs, and microgrids, in providing energy and ancillary services to the utility grid in both normal and emergent status [89]. Compared with conventional stand-by units, the demand-side

resources hold the merit of high flexibility because they are free from ramping constraints. Their diversity in type adds additional reliability for serving as alternative power and frequency support to the bulk power system in case of contingency.

The increasing penetration of demand-side resources into the power system calls for demand-side energy management, which aims to enable a coordinated and mutually beneficial interaction between the main grid and the local resources. One of the primary goals of demand-side management is to reduce the PAR of the load. A low PAR indicates a smooth load profile, which avoids overloading or underloading the system. Local consumers also benefit from a low PAR by shifting their energy consumption to off-peak hours with lower prices.

There have been substantial efforts to investigate the optimal scheduling of demand-side resources in the literature. The concept of autonomous demand-side management is first introduced in [90], in which a non-cooperative game is formulated between the utility company and local customers. Iteratively, the utility provides dynamic pricing signals according to the aggregated consumer response, and the customers optimize their energy consumption schedules under the given price in a distributed manner. At the point of Nash Equilibrium, the minimum total energy cost and the decreased PAR is achieved. In [91], the temporally coupled constraints of the local consumer's energy scheduling problem are included, and the coupled-constrained game model is tackled by dual decomposition. In [92], the authors prove that the non-cooperative game between the users and the utility provider is the general case of the minimum PAR ratio problem. In [93]-[94], the gradient method is utilized for solving a local consumption schedule problem with fast convergence. In [95], an online learning algorithm is developed, where each user learns through past experience to approximate other users' decisions, and to optimize its own energy scheduling.

All of the above methods can be categorized as model-based methods, where the mathematical equations are formulated to describe local users' energy scheduling. Because the demand-side management problem is usually a partially observable problem, i.e., unknown or uncertain

information exists, the models are generally solved in an iterative way. There are two deficiencies of the iterative algorithm: 1) the convergence of the algorithm cannot always be guaranteed. The convergence can only be achieved under some strict prerequisites, e.g., convex payoff functions, which require certain assumptions and simplifications of the problem; 2) applying an iterative algorithm in the real-world can be impractical, especially in real-time scenarios. In real-world practices, it is more likely that the utility provider releases the price signal, and the consumers schedule their consumption accordingly, which tends to be a one-step process. The iterative interaction between the two sides can be both time-consuming and resource-consuming with the potential challenge of divergence.

Based on the above challenges and motivations, we propose a data-driven method in this chapter for optimizing demand-side energy management. Specifically, we propose the combination of two techniques, the DNN and the RL method to overcome the complexity and inefficiency of model-based methods. The recent years have witnessed the rapid advancement of DNN in a variety of applications, e.g., computer vision, machine translation, and remote sensing. In the field of power system, the DNN has been applied for prediction of uncertain factors [96], smart meter data identification [16], modeling of renewable energy [37], and energy storage dispatch [41]. The DNN is a data-driven method that does not rely on any analytical equations, but it utilizes voluminous existing data to formulate the mathematical problem and to approximate the solutions. The multiple hidden layers and the large number of neurons within the DNN can automatically extract features for data analysis to achieve an accurate model regression or classification. Once the DNN is well trained, it will develop high generalization and can be directly applied to new instances without costly numerical computation. Compared to the conventional model-based method, the DNN is highly computational efficient while maintaining considerable accuracy.

The RL method is well known for its applicability in solving problems with hidden information. RL focuses on providing the optimal time-sequential decisions within an unknown environment. This is realized via continuous interactions between the decision-maker, which is called the agent,

and the environment. Through this learning process, the agent is able to gain knowledge of the environment and to take actions that affect the environment in order to reach its objective. Currently, RL has been widely spotted in areas including robotics and automation, computer games, auto pilot, and dialog system.

There have also been significant efforts to implement RL method in solving complex power system problems. The utilization of RL to optimize the residential demand response schedule is first discussed in [97]. The method is later decomposed to the device-level to achieve higher computational efficiency [98]. The research in [99] further includes the smart energy hub to the residential DR management to initiate a real-time energy monitoring and to boost the learning process. In [39]-[40], both a DNN and RL are leveraged for an economically efficient residential load control. DNN is used to estimate the potential reward of each move of the consumer, and RL is used to coordinate the actions from a long-term perspective. This combination is called deep reinforcement learning (deep RL). The authors in [43] proposed the application of deep RL to optimize the real-time electric vehicle charging schedule with the consideration of future electricity price. The feasibility of applying deep RL to load frequency control with stochastic renewable energy penetration is investigated in [45]. More potential applications of deep RL in power system studies have been discussed in [47].

Inspired by the previous works, in this chapter, we propose the utilization of both DNN and RL method to solve the problem of MMG energy management. Different from the load control model in the previous works, a microgrid contains both generation and consumption units, leading to more variables and constraints with higher model complexity. In such cases, the conventional model-based method may become inapplicable due to the computational burden, which makes the data-driven method a more desirable and efficient alternative solution. The main contributions are summarized as follows:

- 1) A data-driven DNN is constructed to model the MMG response under dynamic retail price signals. The DNN is trained based on historical data and without requiring the user information

from local microgrid operators. Uncertain factors within the microgrid system are also included in the training set. The well-trained DNN has high generalization and can automatically generate MMG power exchange under the new given input.

2) A model-free RL technique is applied for the distribution system operator (DSO) to optimize the retail pricing for local microgrids. The RL method aims to maximize the profit of selling power while reducing the PAR ratio. The DSO is able to achieve a near-optimal pricing strategy with the substantial exploration ability of the proposed RL method.

3) A comprehensive performance evaluation of the proposed method is provided through various simulations to verify its feasibility in practical scenarios. A comparison with model-based method is also presented to demonstrate the superiority of the proposed data-driven method.

The rest of this chapter is organized as follows: section 4.2 presents the mathematical model of the MMG energy management problem; section 4.3 demonstrates the detailed design of the proposed DNN and the training process; section 4.4 elucidates the model-free RL algorithm for retail price setting of DSO; section 4.5 provides the simulation results of the proposed algorithm as well as observations and analysis; finally, section 4.6 concludes the chapter.

4.2 Modeling of Multi-microgrid Energy Management

In this section, we first introduce the mathematical model of the proposed MMG energy management problem. The interaction between the MMG and distribution system is shown in Figure 4.1. In the figure, a bi-directional communication channel is constructed between the microgrids and the DSO, where the DSO releases its retail price to the microgrids, and the microgrids send back the amount of power to purchase. The goal of MMG energy management is to smooth the hourly power exchange profile of the MMG with proper retail price setting strategies.

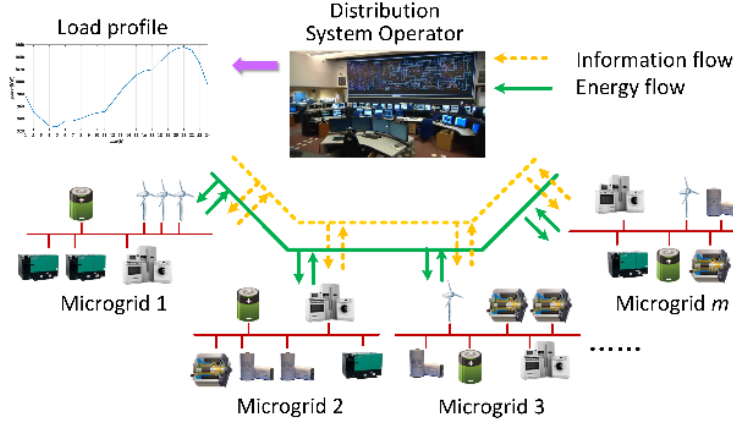


Figure 4.1. Multi-microgrid energy management under DSO pricing control

From the perspective of an individual microgrid, each microgrid operator attempts to minimize its operation cost under the given retail price, which leads to the following microgrid economic dispatch (ED) model:

$$\text{Min} \sum_{t=1}^{N_T} \left(\sum_{k \in m} C_{DG}^P(P_k^{DG}(t)) + \eta_m \lambda(t) P_m^{grid}(t) + \sum_{z=1}^Z ec_m^z q_m^z(t) u_m^z(t) + \rho_{es} |SOC_{es}(t) - SOC_{es}(t-1)| \right) \quad (4.1)$$

$$C_{DG}^P(P_k^{DG}(t)) = a_k^p + b_k^p P_k^{DG}(t) + c_k^p (P_k^{DG}(t))^2 \quad (4.2)$$

The objective function (4.1) represents the operation cost of the m^{th} microgrid over dispatch cycle N_T , which is usually 24 hours. The first term in (4.1) is the generation cost of the k^{th} dispatchable generator, which has a quadratic form of the generation quantity $P_k^{DG}(t)$, as shown in (4.2). The second term in (4.1) is the power exchange cost, where $\lambda(t)$ is the retail price at the point of common coupling (PCC), and η_m is a factor to represent network losses. $P_m^{grid}(t)$ is the power purchased by the microgrid. Note that η_m can differ among different microgrids, because the locations of the microgrids within the distribution network may vary. Thus, each microgrid bears different network losses and receives different retail prices, which is also known as distribution locational marginal price (DLMP). The third term in (4.1) is the cost of dispatching DR resources that reside in the microgrid, where $u_m^z(t)$ is a 0-1 binary variable indicating whether the z^{th} demand

response block $q_m^z(t)$ is dispatched or not, and ec_m^z is the unit price [100]. And the last term is the degradation cost of energy storage. The change between two consecutive states of charge (SOC) is measured as the energy storage life degradation caused by charging or discharging [101].

Microgrid economic dispatch should also satisfy the following constraints:

$$P_k^{DG,\min} \leq P_k^{DG}(t) \leq P_k^{DG,\max} \quad (4.3)$$

$$0 \leq \sum_{z=1}^Z q_m^z(t) u_m^z(t) \leq P_m^{Load}(t) \quad (4.4)$$

$$u_m^{z-1}(t) \geq u_m^z(t), \text{ for } z = 2, \dots, Z \quad (4.5)$$

$$0 \leq P_{es}^{ch}(t) \leq P_{es}^{ch,\max}, 0 \leq P_{es}^{dis}(t) \leq P_{es}^{dis,\max} \quad (4.6)$$

$$SOC_{es}(t) = SOC_{es}(t-1) + \eta_{es} P_{es}^{ch}(t) \Delta - P_{es}^{dis}(t) / \eta_{es} \Delta \quad (4.7)$$

$$SOC_{es}^{\min} \leq SOC_{es}(t) \leq SOC_{es}^{\max} \quad (4.8)$$

$$P_m^{Load}(t) - P_m^{grid}(t) - \sum_{k \in m} P_k^{DG}(t) - \sum_{es \in m} (P_{es}^{dis}(t) - P_{es}^{ch}(t)) - \sum_{z=1}^Z q_m^z(t) u_m^z(t) = 0 \quad (4.9)$$

Constraint (4.3) is the generator capacity constraint of distributed generators (DGs) in the m^{th} microgrid; constraints (4.4)-(4.5) mean that the total demand response dispatched should not exceed the load $P_m^{Load}(t)$, and the demand response blocks are dispatched in an increasing order; constraint (4.6) is the charge/discharge rate limit of the energy storage, where $P_{es}^{ch}(t)$ and $P_{es}^{dis}(t)$ are the charging and discharging quantity of the energy storage; constraint (4.7) calculates the energy level of energy storage, which is $SOC_{es}(t)$, where η_{es} is its efficiency and Δ is the length of the time interval; constraint (4.8) is the capacity limit of energy storage; and finally, constraint (4.9) is the power balance constraint of the microgrid.

The DSO decides the retail price by solving the following optimization problem:

$$\text{Max } \alpha \left(\sum_{t=1}^{N_T} \lambda(t) \sum_{m=1}^{N_m} \varepsilon_m P_m^{grid}(t) \right) / \text{profit}_{base} - (1-\alpha) P_{\max}^{grid} / P_{\text{avg}}^{grid} \quad (4.10)$$

$$P_{\max}^{grid} \geq \sum_{m=1}^{N_m} \varepsilon_m P_m^{grid}(t), \text{ for } t = 1, \dots, N_T \quad (4.11)$$

$$P_{avg}^{grid} = \sum_{t=1}^{N_T} \sum_{m=1}^{N_m} \varepsilon_m P_m^{grid}(t) / N_T \quad (4.12)$$

In (4.10), the first term is the DSO's profit from selling energy to the microgrids, where N_m is the total number of microgrids. ε_m is a conversion factor. This is because $P_m^{grid}(t)$ is calculated by the local microgrid operators and does not include the network losses, hence cannot reflect the real amount of power exchange at PCC. The function ε_m is to transform the local power exchange to the power exchange at PCC. For the sake of simplicity, we do not consider the detailed distribution network topology for a full-fledge DLMP model and assume that ε_m is a known value in the following simulations.

The second term in (4.10) is the PAR over the entire dispatch cycle, which is the ratio between the maximum power exchange and the average power exchange of MMG. Since PAR is unitless, the first term is divided by a constant base $profit_{base}$ value to remove its unit. The DSO intends to find the optimal retail price $\lambda(t)$ that maintains a balance between the two objectives, hence there is a weighting factor α added before the two terms.

The difficulty of solving (4.10) is that the individual microgrid power exchange $P_m^{grid}(t)$ varies with the retail price $\lambda(t)$, hence it cannot be solved directly. In the following sections, we will introduce two data-driven techniques, the DNN and RL, to crack the above problem with high computational efficiency.

4.3 Multi-microgrid Operation Simulation with Deep Neural Network

In this section, a DNN is applied to simulate MMG operation under given price signals, i.e., to solve (4.1)-(4.9). There are two main advantages of utilizing the DNN:

- 1) The neural network is readily available as a toolbox. Once the parameters are well-trained, it has high generalization and can automatically generate the estimated amount of power exchange between the MMG and DSO under the new retail price. Given that the individual microgrid economic dispatch model is a nonconvex problem and that the number of microgrids can be large, solving the MMG power exchange using the conventional analytical method can be highly time-

consuming. The data-driven DNN has much higher computational efficiency with considerable accuracy;

2) The individual microgrids do not need to expose their generation or consumption information to the DSO, given that the DNN is trained using the historical retail price data and power exchange data. Therefore, the user privacy of microgrid owners is well protected.

4.3.1 Deep neural network structure

The ANN has long been recognized as an efficient regression tool for handling problems that are difficult to accurately model or with high computational complexity. MMG energy management fits this category. Hence, a DNN is constructed in Figure 4.2:

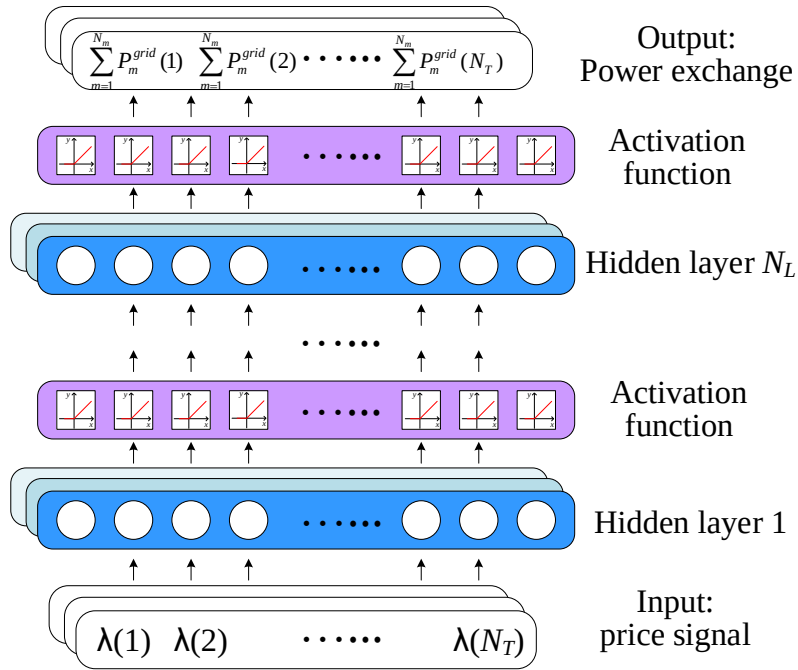


Figure 4.2. Multi-layer structure of the Deep Neural Network

As shown in Figure 4.2, the input to the DNN is the retail price, and the output is the aggregated MMG power exchange with the distribution system under the given price signal. The goal of the DNN is to generate a simulated power exchange that is as close as possible to the actual MMG response.

Before sending the raw training data to the DNN for regression analysis, data preprocessing is implemented. The function of data preprocessing is to minimize the deviation of the training data for improving the regression accuracy and computational efficiency.

The data preprocessing for MMG response raw data includes two steps: firstly, all the sample input data and output data are transformed into the per unit value. By utilizing the per unit value, different features of the sample data become comparable with each other. For the retail price sample, given that they are at the scale of 10\$/MWh, 100\$/MWh is set as the base value; for the aggregated MMG power exchange, given that they are at the scale of 100 kW, 1000 kW is set as the base value.

Secondly, a *min_max_scaler* transformation is applied for further normalization, as shown below:

$$\lambda_s^{new}(t) = \frac{(\lambda_s(t) - \min_s \lambda(t))}{(\max_s \lambda(t) - \min_s \lambda(t))} \quad (4.13)$$

In (4.13), s is the index of training samples, $\max_s \lambda(t)$ and $\min_s \lambda(t)$ are the maximum and minimum values of the retail price at the t^{th} interval among the entire training set. Through the above normalization, the values of the retail price samples will lie within the range of [0,1]. The above data preprocessing helps create a more regular search region for faster algorithm convergence.

In the DNN structure, between the input layer and the output layer are numerous hidden layers. The term “deep” refers to the multiplicity of hidden layers. Each hidden layer is composed of neurons that complete the following affine transformation of the input:

$$y_{sk}^{(l)} = \sum_{j=1} x_{sj}^{(l)} \omega_{jk}^{(l)} + b_k^{(l)} \quad (4.14)$$

The calculation of the output of the l^{th} hidden layer is shown by (4.14), where s is the index of

the sample, j is the index of the features of the sample, and k is the index of neurons. Also, $\omega_{jk}^{(l)}$ is the weight assigned to the j^{th} feature of the input, and $b_k^{(l)}$ is the bias. As can be observed, the output $y_{sk}^{(l)}$ is the weighted aggregation of all the features of the input $x_s^{(l)}$ captured by the k^{th} neuron. The function of the hidden layer is to extract sufficient features from the input data and to construct the mapping between the input and the output.

Notice that (4.14) is a linear transformation. However, the microgrid ED model (4.1) is nonlinear, and cannot be handled by a mere linear transformation. The ReLU function is thus added to the hidden layer to delinearize the model, as shown in Figure 4.2.

4.3.2 DNN training algorithm

In the DNN, the network parameters $\omega_{jk}^{(l)}$ and $b_k^{(l)}$ are the unknown variables that need to be calculated. The back-propagation algorithm is applied for this cause. Before the implementation of the algorithm, a loss function is defined as the objective of the DNN training. The loss function implies the accuracy of the output from the DNN. In the MMG energy management problem, mean square error (MSE) is utilized as the loss function:

$$L(\omega, b) = \frac{1}{N_S N_T} \sum_{s=1}^{N_S} \sum_{t=1}^{N_T} (P_{total,s}^{grid*}(t) - P_{total,s}^{grid}(t))^2 \quad (4.15)$$

$$P_{total,s}^{grid*}(t) = \sum_{m=1}^{N_m} \varepsilon_m P_{m,s}^{grid*}(t), \quad P_{total,s}^{grid}(t) = \sum_{m=1}^{N_m} \varepsilon_m P_{m,s}^{grid}(t) \quad (4.16)$$

In (4.15), N_S is the number of training samples, $P_{total,s}^{grid*}(t)$ is the actual MMG power exchange at the t^{th} time interval of the s^{th} sample, $P_{total,s}^{grid}(t)$ is the estimated MMG power exchange. The loss function tries to minimize the deviation between the ground truth and the estimated value to obtain an accurate enough approximation of the MMG response.

In the studied MMG system, there exist uncertainties, e.g., distributed renewable generation fluctuation, load variations. These uncertainties may cause extremely large or small power exchanges. The existence of such abnormal values in the training set can lead to the issue of overfitting, where the DNN attempts to fit to all the training samples and loses its generalization.

To overcome the overfitting problem, we introduce L_2 regularization to the loss function (4.15), which is shown as follows:

$$L(\omega, b) = \frac{1}{N_S N_T} \sum_{s=1}^{N_S} \sum_{t=1}^{N_T} (P_{total,s}^{grid*}(t) - P_{total,s}^{grid}(t))^2 + \frac{\alpha}{2} \omega^T \omega \quad (4.17)$$

Once the loss function is calculated, the first partial derivatives of the loss function to the weights and biases can be obtained and used to update the variables.

4.4 Monte Carlo Reinforcement Learning Method for DSO Decision-making

In section 4.3, the DNN is constructed to simulate the MMG operation under the given price. As such, the DSO can obtain a reliable estimation of the aggregated MMG power exchange without much computation. Next, the DSO will decide the optimal retail price setting with the goal of maximizing the profit of selling power and minimizing the PAR, as shown by (4.10).

Note that the PAR in (4.10) is not an explicit expression of the decision variable, which is the retail price $\lambda(t)$, hence (4.10) is difficult to solve. In previous literature, similar problems are usually solved in a distributed and iterative manner, where the utility provider first releases the retail prices, and each local user sends back their power consumption under the given price. The utility provider then evaluates the current PAR and adjusts the price accordingly. The above process repeats until no power consumption change or price change happens.

The iterative method is not applicable to MMG energy management problem for the following two reasons: 1) in previous studies, the local users are only consumers and are only allowed to shift their load. In this way, the total energy demand becomes constant and the average hourly load can be calculated, which only leaves P_{max}^{grid} unknown, as is the case in [90],[93]. However, in the MMG case, since each microgrid is a prosumer, their final energy consumption cannot be predicted, hence both P_{max}^{grid} and P_{avg}^{grid} are unknown terms, and the division leads to a nonconvex problem, the iterative algorithm cannot guarantee to converge in a nonconvex case; 2) The iterative algorithm can be time-consuming and resource-consuming, and not applicable for real-time applications.

Motivated by the above considerations, the RL method is applied in this study to crack the intractability of the DSO pricing problem. The RL method is well-known for its applicability to problems with unknown search spaces. For example, in the MMG energy management problem, both maximum power exchange P_{max}^{grid} and average power exchange P_{avg}^{grid} remain unknown to the decision-maker DSO, and they are also not analytically expressed as functions of the retail price. The RL method has strong exploration abilities through continuous interactions with the unknown environment and constantly updates the agent's experience in order to make the optimal decision. In this section, we will discuss how to implement the RL method to optimize the retail pricing strategy of DSO.

4.4.1 A brief overview of reinforcement learning

RL is a type of machine learning approach focusing on how agents take actions within an unknown environment with the goal of maximizing reward [102]. Briefly speaking, in a provided environment, at each state, the agent randomly takes an action, and receives an immediate reward from the environment. Then the agent moves to the next state with a certain probability and repeats the above process, as shown in Figure 4.3:

In the beginning, the agent has no knowledge of what reward and next state are linked to each action. To maximize the accumulative reward, the agent must learn the above knowledge by continuously interacting with the environment. In most cases, the action taken at the current state not only affects the immediate reward, but also the next state and all the future rewards. Hence, it can be concluded that RL is a decision-making process with trial-and-error-search and delayed reward.

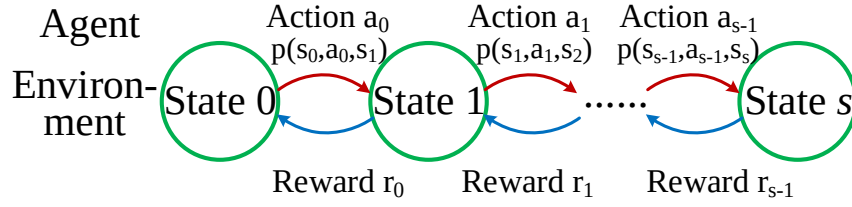


Figure 4.3. Illustration of reinforcement learning

4.4.2 Mapping multi-microgrid energy management problem to reinforcement learning

RL assumes that the problem under study is a Markovian Decision Process (MDP), which is composed of four fundamental elements: 1) a series of environment states $\mathcal{S}$; 2) a set of actions $\mathcal{A}$; 3) a sequence of rewards $\mathbf{R}$; and 4) the probability $\mathbf{P}$ that describes the transition from state s and action a to state s' and reward r .

In the MMG energy management problem, the fundamental elements of the RL are defined as follows:

- The agent: DSO
- State: current time step t
- Action: hourly retail price $\lambda(t)$, for $t = 1 \dots, N_T$
- Reward: Hourly profit of selling power, $\lambda(t) \sum_{m=1}^{N_m} \varepsilon_m P_m^{grid}(t)$

The ultimate objective of DSO is to maximize the total profit of selling power over the entire dispatch cycle, plus weighted PAR, as shown in (4.10). Since both the accumulative profit and PAR are decided by the power exchange through the entire dispatch cycle instead of a single time step, the DSO has to be farsighted to predict the future MMG power exchange when deciding the retail price for the current time step. This corresponds with the delayed-reward feature of RL, and makes RL a natural fit for the MMG energy management problem.

Note that the transition function is not given in the above definitions. This is because the reward in this problem is related to the hourly total power exchange of MMG, which is difficult to predict. The hourly power exchange of microgrids is related to various uncertain factors within the

microgrid system, e.g., load variations and distributed renewable energy. In the next subsection, we will introduce a model-free method to overcome the barrier of lacking transition function.

4.4.3 Model-free Monte Carlo method

There are two types of RL methods: the model-based method and the model-free method. The former assumes that the problem is a known MDP with full knowledge of state transition probabilities. In this way, the problem can be solved analytically via dynamic programming or other iteration methods. However, for some RL problems, obtaining the transition probabilities is not a trivial task. In such occasions, the agent has to estimate the transitions and rewards from the interactive experiences with the environment. This is called the model-free method, since no state transition model can be constructed in advance due to the lack of information.

The Monte Carlo method is a type of model-free method. To obtain the state and the reward information, the Monte Carlo method deploys the simplest possible policy. It utilizes the averaged sample reward for a certain action as its reward value. According to the law of large numbers, when there are enough simulations and enough samples of reward, the averaged value is approximately equal to the actual value, which proves the reasonability of the Monte Carlo method.

As mentioned previously, it is very difficult to obtain the transition probability of the state and reward, which involves the hourly total power exchange of MMG, mainly due to the microgrid uncertainties. Therefore, in the MMG energy management problem, we also adopt the model-free Monte Carlo method to optimize the retail pricing strategy of the DSO. The Monte Carlo method is displayed in Algorithm 1 [102]:

Algorithm 1: Monte Carlo Method for DSO decision-making

```
1: Generate daily retail price sequence samples  $N_S$ 
2: Input the price samples to the DNN to obtain the MMG power
   exchange profile
3: for  $t$  in 1 to  $N_T$  do
4:     Choose retail price  $\lambda^{(s)}(t)$  from price samples  $N_S$ 
5:     Initialize the counter  $n(s) \rightarrow 0$ 
6:     for  $s'$  in 1 to  $N_S$  do
7:         if  $\lambda^{(s')}(t)$  equals  $\lambda^{(s)}(t)$ 
8:             do  $n(s) \rightarrow n(s) + 1$ 
9:         end if
10:    end for
11:    Evaluate  $\lambda^{(s)}(t)$  based on average weighted reward:
         $r(\lambda^{(s)}(t)) = 1/n(s) \cdot (\alpha \sum profit(\lambda^{(s)}(t)) - (1-\alpha) \sum PAR(\lambda^{(s)}(t)))$ 
12:    Select  $\lambda(t) = \argmax r(\lambda^{(s)}(t))$ , for all  $s \in N_S$ 
13: end for
```

Algorithm 1 is explained as follows: to begin with, the DSO randomly generates large quantities of retail price sequence samples. The price samples are then sent into the DNN to obtain the estimated aggregated MMG power exchange. After the generation of all the price samples and the power exchange samples, the DSO selects the optimal hourly retail price based on the procedure as follows:

First, at each time step t , the DSO randomly picks a retail price $\lambda^{(s)}(t)$ from the sample set, then counts the number of price samples that contains $\lambda^{(s)}(t)$ and records it as $n(s)$.

Then, the DSO evaluates $\lambda^{(s)}(t)$ based on its average profit and average PAR. The $profit(\lambda^{(s)}(t))$ is calculated as follows:

$$profit(\lambda^{(s)}(t)) = \sum_{k=t}^{N_T} \gamma^{k-t} \lambda^{(s)}(k) \varepsilon_m P_m^{grid,s}(k) / profit_{base} \quad (4.18)$$

It can be seen from (4.18) that the profit of $\lambda^{(s)}(t)$ is the discounted accumulated profit of selling power from t to N_T , where the discount factor γ is between 0 and 1. When γ is zero, it implies that the decision-maker focuses only on the current profit and is totally myopic; when γ is greater than

zero, it means that the decision-maker is farsighted by evaluating the current pricing with the consideration of potential future profit. In this study, γ is set to 0.9 to ensure that the DSO has a more robust pricing strategy to avoid future risks.

Next, for a single price $\lambda^{(s)}(t)$, the PAR under the price sequence $[\lambda^{(s)}(1), \dots, \lambda^{(s)}(t), \dots, \lambda^{(s)}(N_T)]$ is taken as $\text{PAR}(\lambda^{(s)}(t))$. Each price is then evaluated based on the weighted sum of average $\text{profit}(\lambda^{(s)}(t))$ and average $\text{PAR}(\lambda^{(s)}(t))$, as shown in line 11 of Algorithm 1. The weight factor α represents the tradeoff between maximizing profit and minimizing PAR.

Finally, all the prices are compared and the price with the maximum weighted reward is selected as the price for time step t , as shown in line 12. The above process is repeated for all the time steps until the whole optimal retail price sequence is decided.

The above algorithm is a Monte Carlo method because the DSO selects the optimal price sequences from a randomly generated sample set. Note that in the above algorithm, the price for each time step is selected separately, i.e., the price selection process (line 6-line 10) repeats for N_T times to obtain a complete price sequence. A more intuitive way is to directly select the price sequence with the maximum weighted reward from the sample set. However, this intuitive method cannot guarantee to reach global optimization when the possible realizations of the price sequence are huge. For instance, if there are N_T time steps in a dispatch cycle, and for each hour, there are N_p possible prices, then the total number of candidate price sequences will be $N_p^{N_T}$, which can be an enormous figure even for small N_p and N_T , and cannot be completely represented by a limited sample set. By using the average value to evaluate each hourly price and regrouping them, the algorithm can explore beyond the given sample set and discover solutions better than the existing combinations. This judgement will be verified in the simulation part in next section.

4.5 Simulation Analysis

In this section, we first reveal the detailed structural design of the DNN for simulating MMG operation. Then the testing performance of the DNN is presented. Next, based on the simulated

results from the DNN, the model-free Monte Carlo method is applied for the DSO to decide the optimal pricing strategy. The results are evaluated and compared with a conventional model-based method to demonstrate the advantages of the proposed data-driven method.

4.5.1 Simulating multi-microgrid Operation with DNN

1) MMG system setup

A test case where 10 microgrids are connected to one DSO is considered here. For simplicity, we assume that microgrids with greater serial number are farther away from PCC, hence suffer from more network losses and receive a higher retail price. The η_m for the 10 microgrids are assumed to be in the range of 1.01-1.1, with an incremental size of 0.01. The setting of η_m is aligned with the results of distributional locational marginal price (DLMP) in [103], in which the DLMP range is around 100% to 110% of the price at PCC. The ε_m is assumed to be the same as η_m . The compositions of each microgrid are summarized in Table 4.1:

Table 4.1 Microgrid composition

No.	Compositions	No.	Compositions
1	WT, DE, DE, ES, DR	2	WT, DE MT, FC,ES,DR
3	WT, MT,MT,FC, ES,DR	4	WT, MT, FC,ES,DR
5	WT, DE, MT, MT,ES	6	WT, DE, FC, FC,ES,DR
7	WT, DE, DE, FC, ES, DR	8	WT, FC, FC, ES, DR
9	WT, DE, MT, FC, FC, ES, DR	10	WT, MT, MT, MT,ES,DR

WT: wind turbine; DE: diesel generation; MT: micro turbine; FC: fuel cell; ES: energy storage; DR: demand response

Table 4.2. Parameters of distributed energy resources

DG type	$P_k^{DG,min}$ (kW)	$P_k^{DG,max}$ (kW)	Quadratic coefficients		
			a_k^p (\$/h)	b_k^p (\$/kWh)	c_k^p (\$/kW ² h)
Micro turbine	0	30	0.4	0.0397	0.00051
Fuel cell	0	30	0.38	0.0267	0.00024
Diesel generator	0	60	1.3	0.0304	0.00104
Energy storage	SOC_{es}^{min} (kWh)	SOC_{es}^{max} (kWh)	$P_{es}^{ch,max}$ (kW)	$P_{es}^{dis,max}$ (kW)	ρ_{es} (\$/MW)
	20	50	25	25	0.9
DR quantity	33% of total DR		66% of total DR		100% of total DR
DR unit price (\$/kWh)	0.44		0.46		0.48

2) Design of DNN regression model

We design a DNN with 3 hidden layers for simulating MMG operation under the given retail price. The number of neurons in each layer is 1000. The number of inputs and outputs are both 24, since there are 24 hourly prices with 24 hourly power exchanges (i.e., the dispatch cycle considered here is 1 day). The number of neurons in each hidden layer is decided via repeated trial and error. The selection of the number of neurons is a trade-off between the regression accuracy and computational efficiency. The self-adaptive Adam Optimizer is applied with an initial learning rate of 1e-2 [104]. In addition, the exponential decay of learning rate is applied to stabilize the training. The initial values of the weights and biases of the DNN are obtained from Xavier initialization [105]. Furthermore, to guarantee that the output from each hidden layer is regularized within a certain range, batch normalization is applied to avoid algorithm divergence [106].

In this case study, 12,000 samples of retail price and power exchange are generated for the neural network training. In the first place, the daily retail price is randomly generated as 1 to 1.5 times higher than the wholesale market price, with a step size of 0.1. The wholesale market price can be obtained from historical market data. Then the generated retail price data is sent to model (4.1)-(4.9) to calculate the hourly power exchange of each participating microgrid. In addition, there exist uncertain factors with the microgrid, e.g., the output of wind turbine, and the demand

variation. To make the DNN regression model more robust against uncertainties, we assume that the forecast error of load and wind generation follows a normal distribution with zero mean and a standard deviation of 0.1 and 0.05, respectively. Because a large number of training samples are generated to cover enough uncertain scenarios, the well-trained DNN has high generalization to unseen microgrid uncertainties and can provide regression results with high accuracy.

3) DNN training and testing results

The conventional model-based method is used at this stage to solve model (4.1)-(4.9). In this study, we use GAMS/CPLEX software package to solve the model. The ratio of training samples to testing samples is 8:2. The total number of iterations is 2,000. The hardware environment is a Nvidia GeForce GTX 1080 Ti Graphic Card with 11 GB memory and 1.582 GHz core clock. The software environment is the online open-source deep learning platform TensorFlow, which is implemented on Python. The whole simulation framework is shown in Figure 4.4.

Table 4.3 shows the detailed settings of DNN training. The training result is summarized as follows: for the 2,400 test samples, the average relative error of estimated power exchange is 0.96%, which indicates the considerable accuracy of DNN regression. The training loss (MSE plus L_2 weighted penalty) for 9,600 training samples is 0.0034, which is small enough as an indicator of the training convergence. The time for completing 2,000 iterations is 91.05 s, which is acceptable since the training is completed off-line.

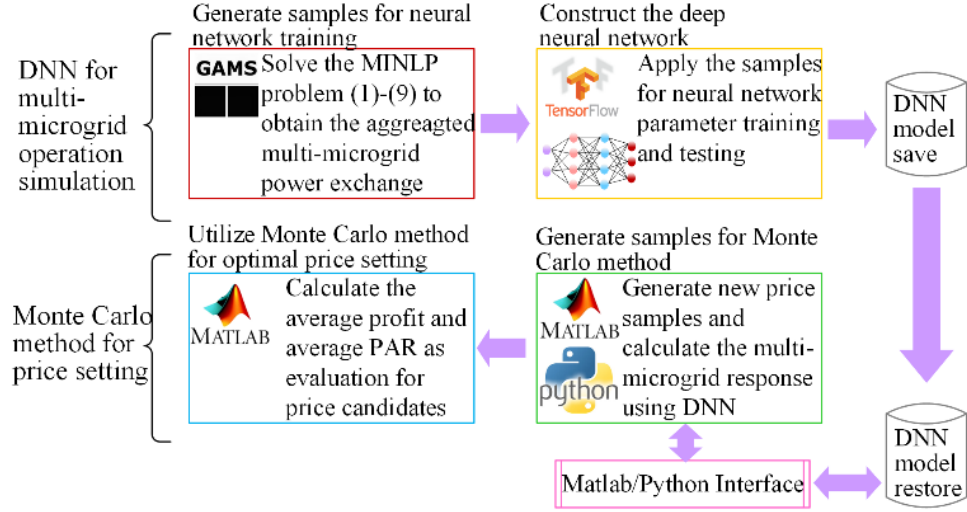


Figure 4.4. Simulation framework for multi-microgrid energy management

Table 4.3 Summary of DNN training settings

Item	Value
No. of hidden layers	3
No. of neurons in each hidden layer	1000
Activation function	ReLU
Loss function	MSE plus L ₂ regularization
Learning rate	1e-2
Exponential decay rate	0.96
Exponential decay step	50
Optimizer	Adam Optimizer
No. of training samples	9,600
No. of test samples	2,400
Iteration steps	2,000
Data preprocessing	min_max_scaler

4) Sensitivity analysis

To further verify the high generalization of the well-trained DNN to unseen inputs, we conduct the following sensitivity analysis of the DNN regression accuracy.

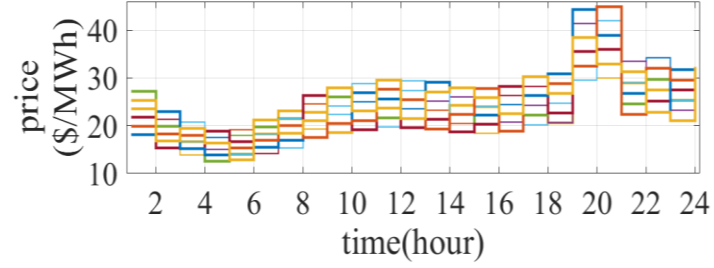
First, the effect of price disturbance is discussed. As previously discussed, for the training set generation, the daily retail price is randomly generated as 1 to 1.5 times higher than the wholesale

market price. To include the price disturbance, in the test set generation, we manually create a price peak at hours 10-11. Note that this disturbance is not included in the training set. The comparison of price samples for training and testing is shown in Figure 4.5.

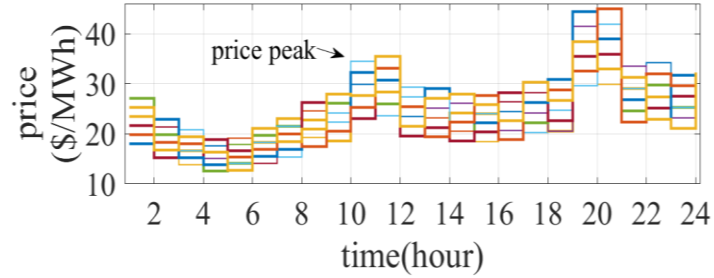
Note that there appear to be only six lines in Figure 4.5, because some price samples overlap with others. A test set with the size of 500 based on the above price disturbance is generated and input to the DNN. The average relative error of estimated power exchange is 1.65%. Note that this error is slightly higher than the above 0.96%, given that the price disturbance is not included in the training set. Still, this average relative error is low enough to verify the robustness of DNN regression under price disturbance.

We further explore the effect of microgrid load variation to the DNN regression accuracy. Similar to retail price disturbance, we manually create a load valley for hours 13-14 on the original microgrid load profile for the test set generation. The comparison of the microgrid load for training and test is shown in Figure 4.6. A test set with the size of 500 based on the microgrid load disturbance is generated and input to the DNN. The average relative error of estimated power exchange is 1.60%. This further verifies the robustness of DNN regression under load profile disturbance.

Based on the above observations, it can be safely concluded that the DNN has formulated a considerably accurate regression model between the input, which is the retail price, and the output, which is the MMG power exchange, and is immune to the unseen disturbance in the input data. This is due to the strong automatic feature extraction ability of the large number of neurons embedded within the DNN. As a result, the DNN has tremendous potential in solving problems with unclear or complex mathematical formulations.

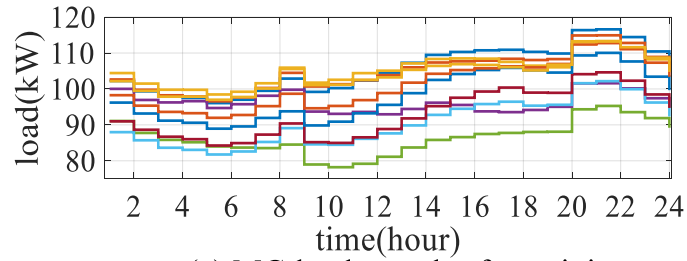


(a) price samples for training

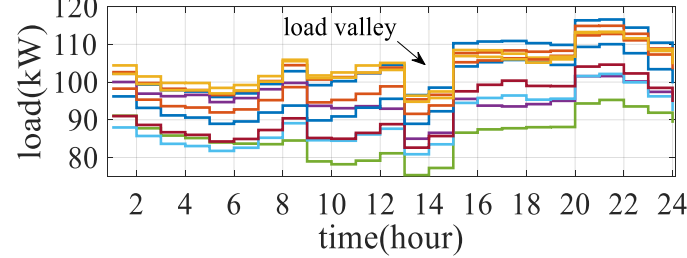


(b) price samples for test

Figure 4.5. Disturbance of retail price



(a) MG load samples for training



(b) MG load samples for test

Figure 4.6. Disturbance of microgrid load

4.5.2 Monte Carlo method for optimizing DSO pricing strategy

Once the DNN is well trained, the fine-tuned parameters can be properly stored for repeated use. The DSO can now apply the Monte Carlo method to search for the optimal retail price for the

MMG. Since the Monte Carlo method is based on the law of large numbers, the more samples are generated, the closer the obtained solution is to the actual global optimum. As previously mentioned, each hourly retail price falls within 1 to 1.5 times of the wholesale market price, with a step size of 0.1. Then the total number of all possible price sequences is $5^{24} \approx 5.96 \times 10^{16}$, which is far beyond the hardware's computation capabilities. Instead, we generate 10^4 , 2×10^4 , 5×10^4 , and 8×10^4 price samples respectively, to observe the effect of sample sizes on the performance of the Monte Carlo method.

In the first place, the computation time for using the DNN to calculate an MMG power exchange, and for the Monte Carlo method to scan all the generated samples for price setting are shown in Table 4.4. The DNN calculation and Monte Carlo method are implemented on Matlab R2017b plus Python, and the hardware environment is a laptop with Intel®Core™ i7-7600U 2.8 GHz CPU, and 16.00 GB RAM. As seen from Table 4.4, using the well-trained parameters of the DNN to calculate the approximated MMG power exchange is fast enough to generate large numbers of samples for the Monte Carlo method. Also, the proposed Monte Carlo method is able to scan through large quantities of candidate retail price sequences with an acceptable time elapse.

In addition, we also test the computing time for solving (4.1)-(4.9) using a conventional model-based method. The software solver is GAMS/CPLEX, and the hardware environment is the same as previously mentioned. The computational efficiency is shown in the last row in Table 4.4. The acceleration ratio is the ratio between the computation time of model-based method and the DNN regression. The latter is thousands of times faster than the former, thus the high computational efficiency of the data-driven DNN is verified.

Note that in Algorithm 1, each price is evaluated by a weighted reward. The value of the weight factor α will affect the eventual price selection. Figure 4.7 demonstrates the optimal price setting obtained by the Monte Carlo method with different weight factors. Figure 4.8 compares the total profit and PAR under different weight factors. More detailed explanations of Figure 4.7- Figure 4.8 are shown as follows.

Table 4.4. Computation time for DNN and Monte Carlo method

No. of samples	Calculation time(s)	
	DNN	Monte Carlo
10,000	2.67	31.35
20,000	3.84	35.25
50,000	7.90	41.82
80,000	12.51	51.44
No. of samples	Calculation time(s) (model-based method)	Acceleration ratio of DNN
10,000	28,301	10,600

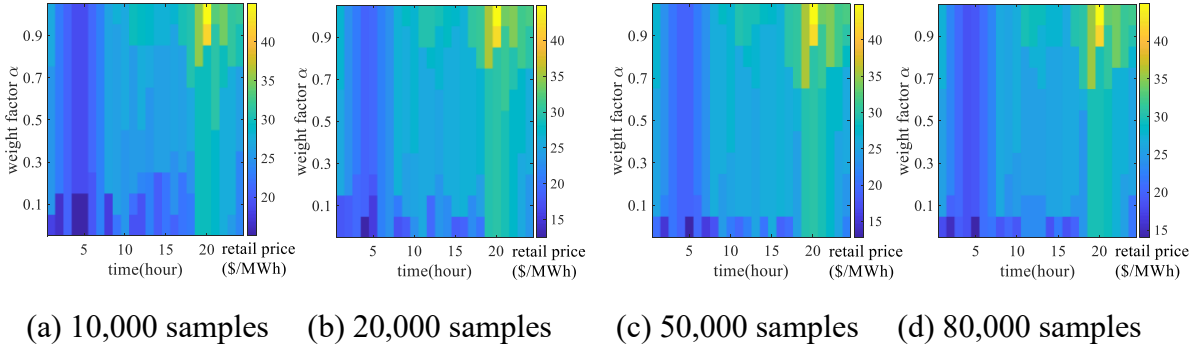


Figure 4.7. Optimal price setting under different weight factors

In Figure 4.7, it can be observed that with a larger weight factor, the DSO intends to increase the hourly retail price. For example, in all the subfigures at hour 20, as α increases from 0 to 1, the hourly retail price goes from green, which stands for a lower price value, to bright yellow, which stands for a higher price value. This is because an increasing weight factor implies that the DSO weighs the profit of selling power more than the PAR, as shown in (4.10). The DSO intends to raise the price to achieve a higher profit.

Figure 4.8 demonstrates the DSO's profit of selling power and the PAR under the specific weight factor. The profit and PAR shown in the figures are obtained by sending the selected price sequence to the individual microgrid model (4.1)-(4.9), and to calculate their aggregated power exchange. The DNN is not used here because we only need to test the selected price sequence, and the conventional model will provide an accurate result. As seen in the figures, a growing weight

parameter leads to higher profit and higher PAR. For example, when α is 0.1, the optimal profits of selling power obtained based on 10,000, 20,000, 50,000, and 80,000 samples are \$609, \$626, \$663, and \$651, respectively, and the optimal PARs are 1.0686, 1.0744, 1.0744, 1.0744; when α is 0.7, the optimal profits of selling power obtained based on 10,000, 20,000, 50,000, and 80,000 samples are \$681, \$679, \$682, and \$682, respectively, and the optimal PARs are 1.082, 1.0777, 1.0840, and 1.0840, respectively. This is because with an increasing weight factor, the DSO values the total profit more than PAR, and tends to increase the hourly retail price, which has already been discussed in Figure 4.7. An increasing price level drives microgrids to shift more of their load to hours with relatively lower prices, which exacerbates the peak to valley distance, and increases PAR. Therefore, the DSO needs to make a trade-off between gaining more profit and maintaining a smooth load profile.

It can also be observed from Figure 4.7-Figure 4.8 that the results based on larger sizes of samples (i.e., 5×10^4 and 8×10^4) don't show much difference. Hence, we can assume that such sample sizes are large enough for the Monte Carlo method to find the optimal solution.

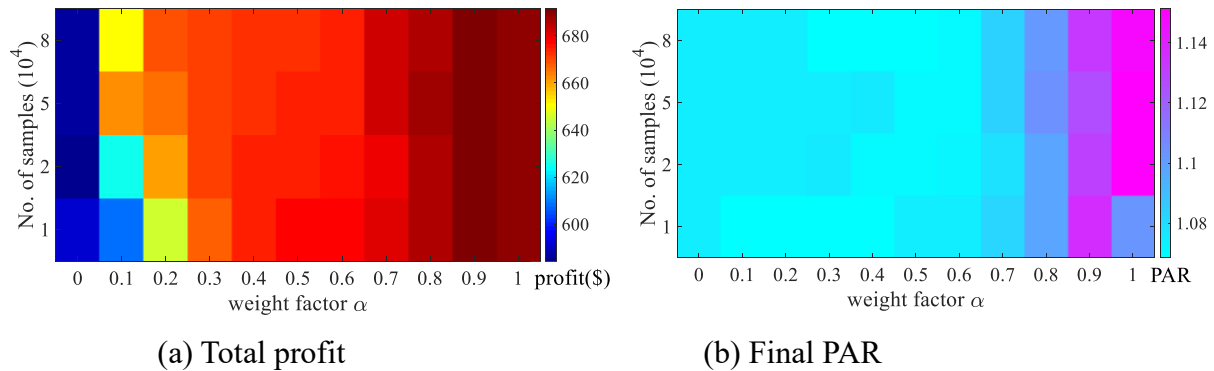


Figure 4.8. Total profit and final PAR under different weight factors

A final conclusion that can be drawn from Figure 4.8 concerns the optimal value of the weight factor. It can be observed from Figure 4.8(b) that as α increases from 0 to 0.7, the PAR increases considerably slow, while the profit of selling power keeps growing. When α is greater than 0.7, the PAR shows an obvious increase. Hence, the DSO is recommended to set the weight factor to 0.7 to maximize the profit of selling power, while maintaining a considerably low PAR.

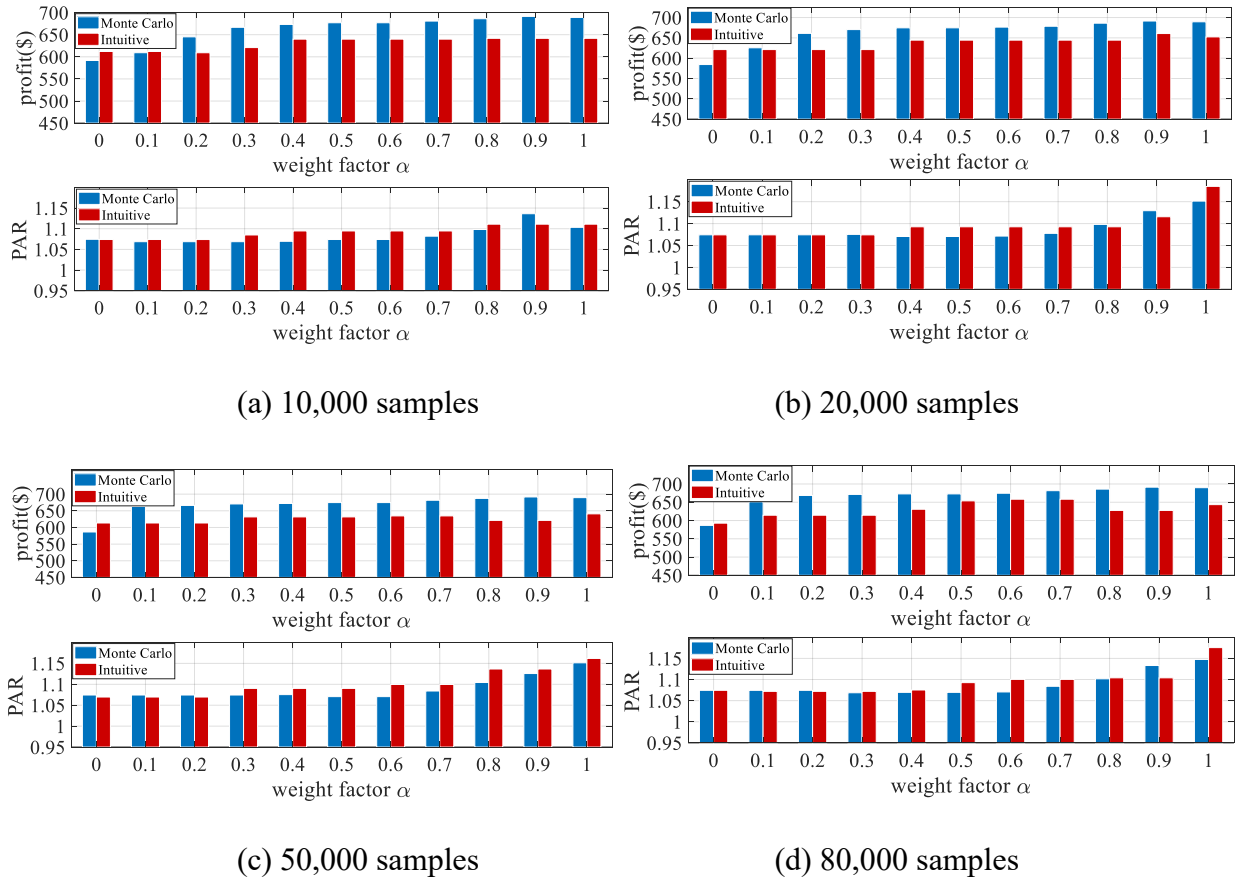


Figure 4.9. Comparison of Monte Carlo method and intuitive method

As stated in section 4.4.3, the Monte Carlo method regroups the prices from different price sequence samples instead of intuitively choosing the price sequence with the largest weighted reward. To verify the merit of the Monte Carlo method, a comparison with the intuitive method is shown in Figure 4.9. As can be observed in the figure, with the change of weight factor, the Monte Carlo method is able to achieve a higher profit of selling power and lower PAR than the intuitive method. For example, in subfigure (d), when α is 0.6, the profits of selling power obtained from the Monte Carlo method and the intuitive method are \$675 and \$658, respectively; and the PARs are 1.0705 and 1.1003, respectively. This is because the Monte Carlo method has a strong exploration ability to discover new price sequences by regrouping the existing price samples, which can lead to better solutions; while the intuitive method only relies on the existing samples, which can be stuck to local optimum.

4.6 Conclusions

In this chapter, a novel data-driven method is proposed for the MMG energy management problem. First, a DNN is constructed to simulate MMG operation under dynamic retail price signals, with no requirement of local generation or consumption information, which protects customer privacy. Second, the DSO applies a model-free Monte Carlo RL method to optimize its pricing strategy, with the aim of maximizing the profit of selling power and minimizing PAR. Simulation results demonstrate that the DNN regression model has considerable accuracy and computational efficiency due to its automatic feature extraction ability and its high generalization. Compared with an intuitive selection method, the Monte Carlo method proves to have strong exploration ability in problems with no explicit mathematical formulations or with high computational complexity. The combination of the proposed data-driven DNN and the Monte Carlo method can be a promising tool for studying power system problems with hidden information or vast search spaces in future researches.

4.7 Acknowledgement

This work was supported in part by NSF Award under Grant ECCS-1809458, and in part by CURENT, a NSF/DOE Engineering Research Center through NSF Award under Grant EEC-1041877.

Chapter 5 Multi-zone Residential HVAC Control using Deep Reinforcement Learning

Residential heating, ventilation, and air conditioning (HVAC) is considered to be an important demand response (DR) resource. Homeowners can greatly reduce their energy cost while maintaining their desired comfort level by optimizing their HVAC control strategy. However, the optimization of the residential HVAC control is not a trivial task due to the complex building thermal dynamics and uncertainty associated with both occupant-driven heat loads and weather forecasts. In this chapter, we apply a novel data-driven multi-zone residential HVAC control method, the deep deterministic policy gradient (DDPG), which belongs to the category of deep reinforcement learning (deep RL), to generate an optimal control strategy of the residential HVAC without referring to any complex modeling formulation or time-costly analytic solving process. The applied deep RL-based method can learn the optimal control strategy through the continuous interaction with the simulated building environment. Simulation results of DDPG on real-world use cases and comparisons with the deep Q network (DQN) as well as with the benchmark cases demonstrate the effectiveness and the generalization ability of DDPG in saving energy cost while maintaining the occupant comfort, which proves its feasibility in solving real-world high dimensional control problems with hidden information or vast solution spaces.

5.1 Introduction

In the worldwide scope, buildings account for 40% of the total primary energy consumption and 30% of all CO₂ emissions, among which a large portion can be attributed to thermal comfort overhead [107]-[108]. Therefore, it is important to study the effective energy management of the building demand to achieve economic and environmental benefits.

The HVAC system is currently the most widely used device for maintaining building thermal

comfort. It also serves as an important DR resource for peak load reduction and stabilizing system-wide operation via proper demand-side energy management strategies [109]. In literature, there are many studies focusing on optimizing HVAC control strategies for improving energy efficiency. In [110], the energy management of HVAC systems is modelled under load forecast errors, where a primal-dual algorithm is applied to seek the optimal operating states of HVAC for the consumer, and the pricing strategy for the energy provider. In another work, a regression approach is applied for temperature forecast for day-ahead scheduling of responsive residential HVAC demand [111]. The authors in [112],[113] discuss the potential of using the HVAC system to provide primary frequency regulation to the bulk system via a hierarchical control strategy. A Lyapunov optimization technique is introduced in [114] for HVAC load control without the need of estimating system's uncertain factors like price and temperature. A distributed transactive control market mechanism for commercial building HVAC systems is presented in [115] to demonstrate the effectiveness of HVAC at peak shaving and load shifting.

All of the above methods can be categorized as model-based methods, where the detailed thermal dynamics of the HVAC with consideration of ambient environmental effects need to be modelled, along with the requirement of analytical solution toolboxes for practical runtime control. The model-based methods may suffer from measurement errors (e.g., building model inaccuracy), as well as computational inefficiency, since the building and equipment models must be tailored to a specific building to achieve accurate results. This represents a serious challenge for widespread deployment of model-based methods.

The smart meter and related technology innovations over the past decade have built a large data repository that enables the application of the data-driven deep learning approaches [116]. The automatic feature extraction ability and generalization ability of deep learning makes it possible to overcome the modeling and computing limitations of the conventional model-based method. In the most recent years, the deep RL, which is a combination of DNN and RL, has attracted broad attention in solving high-dimensional control and optimization problems with tremendous

complexity. Both academia and industry have witnessed the near-human or superhuman performance achieved by the deep RL agent in problems like game of Go [1] and Atari [117]. In the field of power and energy, a double Q learning method [118] and a continuous DDPG method [119] have been applied for optimizing the energy management strategies of hybrid electrical vehicles (HEVs), respectively. In [120], the asynchronous advantage actor-critic (A3C) is employed to find the economic operation schedules of multiple distributed energy resources (DER) within an energy Internet. In [121], a deep Q learning method is designed for supporting the maintenance decision-making of the bulk power system. Given the potential operation constraints encountered during the implementation of deep RL-based control actions, a safe deep RL method is explored in [122] to obtain the optimal control scheme of active distribution network (ADN) with the consideration of voltage level limits, which introduces a safe layer on top of the conventional actor network to avoid any possible violations of the voltage constraints.

With specific respect to the HVAC system control problem, there have also been some pioneering works in the literature focusing on utilizing the powerful deep RL approach to achieve higher energy efficiency and economic efficiency. In [123], a DQN is constructed for coordinated control of joint datacenter and HVAC load, in which the neural network is utilized to estimate the Q value of state-action pair. In [39], a CNN is deployed as the approximator of the state-action value function to better capture the spatial and temporal correlations within the input state data with its convolutional operation. A deep policy gradient (DPG) method is investigated in [40] for controlling multiple responsive demand including ACs, electrical vehicles and dish washers. In [124], an actor-critic method is applied for optimizing the thermal comfort and energy consumption of HVAC.

All of the above existing research works have demonstrated the effectiveness of the applied deep RL methods in optimizing the HVAC thermal control strategy comparing with the designed benchmarks. However, one common deficiency of the above methods is that they cannot handle continuous control actions, like HVAC setpoint or air flow rate. In such cases, the discretization is

mostly applied to partition the continuous action space. Discretization can achieve satisfying performance when the granularity is low or without combination of action spaces. However, it encounters the issue of exponential explosion when the action space is high-dimensional, for example, multiple room zones in the case of HVAC control. As a result, more simulations are needed for training the deep RL methods and the algorithm performance can decrease.

In [125], the authors adopt the DDPG method to realize the continuous thermal control of HVAC without discretization. However, this research work still focuses on single-zone HVAC control, which has been previously addressed by the above-mentioned discretion methods. In addition, the method applied is only compared with other RL methods, and no benchmark cases are designed to verify the optimality and the generalization of the obtained control strategy.

Motivated by such concerns, in this chapter, we also apply the DDPG method for optimizing the continuous thermal control strategy of residential HVAC. The main contributions of this work, if compared with the existing researches, are summarized as follows:

- We apply the DDPG RL method to optimize the continuous control of multi-zone residential HVAC. The multi-zone residential HVAC control involves more complex thermal dynamics and environmental uncertainties, and a high-dimensional action space, which requires more delicate problem formulation including the definitions of state, action, and reward during the learning process;
- We conduct a comprehensive comparison between the applied DDPG method and the widely-used DQN method to demonstrate the effectiveness of the former in dealing with the continuous action space, which is a more common case in many real-world situations; we also design benchmark cases without RL to prove that the applied DDPG can achieve higher economic benefits while maintaining user comfort;
- We verify that the well-trained deep RL method has obtained high generalization and robustness, and is able to adapt to new environment with different price signals and physical conditions to provide the optimal HVAC control strategy.

The rest of the chapter is organized as follows. The HVAC control problem formulation is introduced in section 5.2; in section 5.3, the two representative deep RL methods, the DQN and DPG methods are first briefly reviewed, followed by a detailed explanation of the DDPG method, which is an extension of the former two; the simulation results of the DDPG method are presented in section 5.4, plus comparison with DQN and benchmark cases; finally, section 5.5 concludes the chapter.

5.2 Multi-zone Residential HVAC System Control Problem Formulation

5.2.1 A brief introduction of the multi-zone HVAC system control problem

In this study, we consider a residential building with multiple zones and the indoor temperature of each zone can be controlled by adjusting the setpoint of the HVAC system. The HVAC system can work in various modes including “Cooling”, “Heating” and “Auto”. The “Auto” model means that the HVAC system can automatically switch between cooling and heating according to the indoor temperature and the assigned setpoint. Whenever there is a difference between the indoor temperature and the setpoint, the HVAC system will be automatically turned on to push the indoor temperature near to the setpoint to maintain the user comfort. Without loss of generality, in this work, we will focus on the case when all zones need heating. The goal of controlling the HVAC system is to minimize the energy cost while keeping the indoor temperature within the user comfort band.

5.2.2 Mapping HVAC control problem to Markov Decision Process

In this subsection, we will formulate the above multi-zone residential HVAC control problem as an MDP, which will later be solved by a model-free deep RL-based algorithm in section 5.3. According to the simplified thermal dynamics model of HVAC in [126], the indoor temperature at the current time interval is only related to the previous state parameters such as the indoor temperature at the previous time interval, and is not affected by indoor temperature at any other time intervals. Therefore, the HVAC control problem can be regarded as a finite Markov process and be solved using the RL method.

An MDP is composed of four essential elements: state (s), action (a), state transition probability (p), and reward(r). In the context of multi-zone residential HVAC control problem, the four elements are defined as follows:

- State: 1) current outdoor temperature $T_{out}(t)$; 2) current indoor temperature $T_{in,z}(t)$ for the all the zones z ; 3) the lower bound of the user comfort level $T_{lower}(t)$; 4) retail price $\lambda^{retail}(t)$, where t is the current time step.

Note that the state parameters include the lower bound of the user comfort level, which changes along the time. This is because we assume that the HVAC users have a time-variable comfort preference. This is reasonable since during the daily work hours when no one is at home, the comfort range of the indoor temperature can be lowered to save the energy cost. The comfort range can be brought back during the off-work hours when the house is occupied.

The state parameters also include the current retail price to realize the pre-heating effect of HVAC. Pre-heating means to set the setpoint of HVAC at a relatively high value when the retail price is low to heat up the indoor temperature in advance, to avoid excessive energy consumption when the outdoor cold occurs at a high retail price.

- Action: the setpoint $Setpt_z(t)$ for the zone z ;

The setpoint of HVAC in each zone is a continuous variable. Given the setpoint, the on/off status of the HVAC unit with a thermostat at each zone obeys the following control logic:

$$HVAC\ status = \begin{cases} 1, & \text{if } T_{in}(t) < setpoint - deadband \\ 0, & \text{if } T_{in}(t) > setpoint \\ \text{remain at the current status,} & \text{elsewise} \end{cases} \quad (5.1)$$

The HVAC model considered in this paper is only utilized for heating. In Eq.(5.1), the deadband is a small temperature span, in which the thermostat will not change its on/off status to prevent short cycles. It can be observed in Eq. (5.1) that if the indoor temperature is above the setpoint, the HVAC will remain off; otherwise, the HVAC will be automatically started to heat up the room to maintain the user comfort.

- Reward: the cumulative energy consumption cost for the control interval, which is defined as follows:

$$r(t) = -\omega_c \sum_{t' = t - \Delta t}^t \lambda^{retail}(t') E_{HVAC}(t') - \omega_p \sum_{t' = t - \Delta t}^t c^{penalty}(t') \quad (5.2)$$

In Eq. (5.2), the first term is the energy cost of the HVAC system, where $\lambda^{retail}(t')$ is the retail price, $E_{HVAC}(t')$ is the power consumption, and Δt is the control interval; the second term is the penalty for user comfort violation, which is calculated as follows:

$$c^{penalty}(t') = \begin{cases} 1, & \text{for } T_{in}(t') < T_{lower}(t') - T_{th} \\ 0, & \text{elsewise} \end{cases} \quad (5.3)$$

In Eq.(5.3), T_{th} is a threshold with a small value. The temperature violation is not counted if the magnitude of the violation is smaller than T_{th} . Given the existence of the deadband within the HVAC system, it is not possible to always keep the indoor temperature at the exact setpoint. The threshold allows for some deviations of the indoor temperature.

Because the reward encloses both the energy cost and the penalty, which leads to a multi-objective function, weight factors are added to the two objectives, which are represented by ω_c and ω_p in Eq. (5.2). The final objective of HVAC thermal control is to minimize the total energy consumption cost plus the penalty over the entire control cycle, which can be written as the cumulative sum of $r(t)$: $\sum_{t=1}^{NT} r(t)$. Therefore, a far-sighted control strategy is needed to prevent against uncertain future circumstances, which leads to a multi-stage decision making problem.

Notice that the state transition probability p is not defined for the above MDP. The state transition probability refers to the probability of transferring to a certain next state after taking action $\text{Setpt}_z(t)$. With a known state transition probability, the MDP is fully observed and the cumulative reward can be analytically solved via model-based dynamic programming or other iterative methods. However, in the HVAC control problem, to obtain an accurate probability model of the state transitions is not a trivial task, because it is difficult to formulate the exact thermal-dynamic model of HVAC buildings. The heat transfer within the building is related to multiple

resistance (R) and capacitor (C) from different building components like the exterior walls, the interior walls and furnishings, as well as the attic, the values of which requires estimation and validation through experimenting. All of these factors can have a significant impact on the temperature response of the indoor air [127]. Furthermore, the indoor temperature is also affected by uncertain external factors such as outdoor temperature, solar irradiance, and wind, which calls for additional modelling and computational efforts. As a consequence, a model-based method is not a robust or adaptive solution for HVAC system optimization.

Driven by the above considerations, in this paper the model-free deep RL method is leveraged to overcome the unobservability in the multi-zone residential HVAC control problem. The model-free RL method does not require any knowledge of the environment or the state transitions in advance. It gradually improves its decision-making strategy by continuously interacting with the environment and receiving feedback. In this way, the forecast errors of uncertain factors, as well as the measurement errors of building thermal mass, can be avoided. More details of the deep RL method will be revealed in the next section.

5.3 DDPG-based Control Strategy for Multi-zone HVAC System

5.3.1 A brief review of deep reinforcement learning methods

The RL method is a type of machine learning method that optimizes the decision-making strategy in MDP. In the RL algorithm, the reward defined in MDP is served as the guideline for algorithm evolution. A large, positive reward will encourage the algorithm to search deep in the current direction, and vice versa. The RL method is especially suitable for handling decision-making problems with temporal constraints or with hidden state space.

There are two main types of RL method: the value-based RL method and the policy-based RL method. The difference between the two methods lies in their action evaluation strategies. The value-based method estimates the Q value of a state-action pair (s,a) , which is the cumulative discounted reward starting from taking action a at state s , and selects the action with the highest Q

value; the policy-based RL method generates the probabilities of all the feasible actions at the current state, and selects the action with the highest probability.

The combination of RL with DNN is called the deep RL method. In deep RL, the DNN is utilized as a regression tool to estimate either the Q value, as in the value-based RL method; or the action probability, as in the policy-based RL method. A general DNN structure for regression in RL is shown in Figure 5.1.

The main advantage of the deep RL method over the conventional RL method is that the application of the DNN makes it possible to achieve high level control for extremely complex problems, such as with continuous state space or action space, without the tabular constraints. Since in the deep RL, a more generalized regression model is established instead of maintaining a concrete Q table to store all the possible action values, as in the case of traditional Q learning. This generalized regression model offers more robust and flexible strategies against unseen states in the case of continuous control. In the following section, we will first introduce the DQN, as a representative of the valued-based deep RL methods; and the DPG method, as a representative of the policy-based deep RL methods. Then, a continuous control method, DDPG, which is a combination of the above two methods, will be explained in detail for solving the optimal multi-zone residential HVAC control problem.

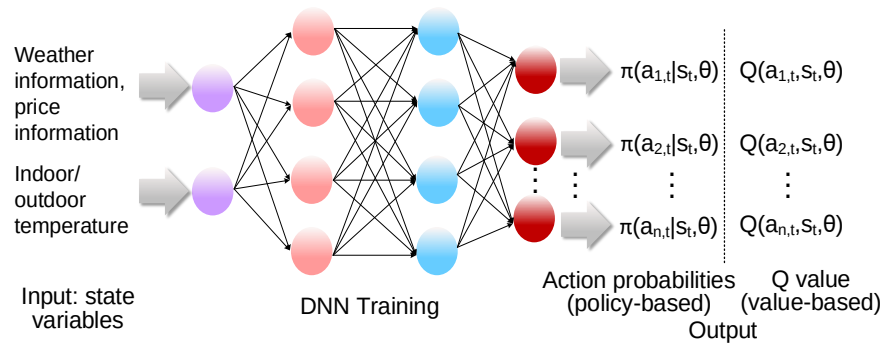


Figure 5.1. DNN structure for function approximation in RL

5.3.2 Understanding the basic principles behind typical deep RL methods

1) Deep Q Network (DQN)

The DQN is a combination of Q-learning and DNN. In the DQN, the input is the current state, and the output is the Q value for each potential action at the current state. The advantage of DQN over the tabular Q-learning method is that once the state and action are slightly changed, the DQN can still estimate the associated Q value without re-training, which is highly time-efficient.

Unlike supervised learning algorithm, in deep RL there are no labeled samples for the DNN to learn. To handle this issue, two DNN are designed for the DQN algorithm: one is called the target network, and the other is called the behavior network. The function of the target network is to serve as a reference, similar to the ground truth in supervised learning, to guide the evolution of the algorithm.

Both networks are initialized with the same parameters and the same structure. As the training proceeds, the behavior network is updated at a faster speed than the target network. The loss function in DQN is defined as the MSE between the target Q value and the behavior Q value. Once the loss function is calculated, the parameters of the behavior network will be updated based on its gradient to the loss function. The algorithm will continue updating until the output from the target network and the behavior network are close to each other, which indicates the convergence of the learning. More details of DQN method can be found in [128].

2) Deep Policy Gradient (DPG)

The DPG method utilizes a strategy different from the DQN for control optimization. The output from the DNN is the probabilities of each potential action at the current state, or the policy. The policy refers to the probability of selecting action $a(t)$ at state $s(t)$, and can be written as $\pi(a|s, \theta) = \Pr\{a(t)=a|s(t)=s, \theta(t)=\theta\}$. θ stands for the parameters of the probability function. The loss function of the DPG method is also different from that of DQN, which intends to maximize the expected total reward under the policy $\pi(a|s, \theta)$, and can be expressed as follows:

$$\max_{\theta} J(\theta) = E_{\pi(a|s, \theta)} \left(\sum_{t=1}^{N_T} r(t) \right) = \sum_{\tau} \pi_{\theta}(\tau) R(\tau) \quad (5.4)$$

In Eq. (5.4), τ is called an episode generated under the policy $\pi(a|s, \theta)$: $\tau = \{s(1), a(1), s(2), a(2), \dots, s(N_T), a(N_T)\}$. $R(\tau) = \sum_{t=1}^{N_T} r(t)$, which is the total reward of the episode. The goal of the DPG method is to get the parameters of the policy π that leads to the maximum value of the expected total reward. More details of DPG algorithm can be found in [40].

5.3.3 Realizing the continuous control of HVAC system with DDPG

1) An introduction to DDPG

The DDPG method is specially designed for solving problems with continuous variables. Unlike DQN or DPG, where the Q values or action probabilities of all feasible actions are generated by the DNN for the agent to select, the term “deterministic” in DDPG refers to the fact that there is only one output from the DNN, which is determined. In this way, the action space can be continuous since there is only one output unit.

Another advantage of DDPG over DQN and DPG is that it is a combination of the two methods. In the DDPG, there are two types of neural networks applied: the actor network, which assembles DPG, and the critic network, which assembles DQN. Their functions are explained as follows.

The input to the actor network is the current state, and the output is a deterministic action; the input to the critic network is the current state plus the action generated by the actor network, and the output is the Q value of the state-action pair. This Q value will be further used to update the parameters of the actor network. The loss function of the actor network is defined to maximize the Q value with the current policy, which follows the logic of the DPG method; and the loss function of the critic network is the MSE of the Q value, which follows the logic of the DQN method. In summary, the function of the actor network is to select actions, and the function of the critic network is to evaluate the selected action.

In addition, similar to the DQN algorithm, for both actor network and critic network in DDPG, two neural networks are designed, a behavior network and a target network. Hence there are four

neural networks in total. The reason for applying a target network is to stabilize the algorithm convergence. More details of the DDPG algorithm are presented in the next subsection.

2) DDPG algorithm for developing optimal HVAC control strategy

The details of the proposed DDPG algorithm are shown in **Algorithm 1**, which is customized from a general DDPG algorithm in [129]. The DDPG algorithm follows a process similar to that of the DQN, except that an actor network is built to select a deterministic action. The proposed DDPG algorithm is further explained as follows:

To begin with, two neural networks, i.e., the actor network and the critic network are randomly initialized, and their associated target networks are initialized with the same set of parameters, as shown in line 1-2. Starting from line 3, for each iteration, the system state is first initialized, then an HVAC control action, i.e. the setpoint, is chosen based on the current actor network $\pi(s; \theta^\pi)$, as shown by line 7. A noise is added to the selected action to boost the exploration of the algorithm.

Next, in lines 8-9, the selected action is executed in the environment for the entire control interval Δt , and the received reward and the next state are observed. The transition $(s(t), \text{Setpt}_z(t), r(t), s(t + \Delta t))$ is stored in a replay buffer to be further used for algorithm training. When a sufficient number of transitions are collected, a mini-batch of transitions is randomly selected to update the parameters of the actor network and the behavior network, as shown by line 11. The random selection can cut off the temporal correlations among the transitions, which will maintain the independent, identically distributed (IID) assumption in the learning model. Also, the transitions can be sampled multiple times, which increases their utilization efficiency.

The neural network parameters θ^Q and θ^π are updated according to the loss functions. The loss function of the critic network is defined as the MSE between the target Q value and the current Q value from the behavior critic network, as shown by line 12. The temporal-difference (TD) error is used to update Q value, where the target Q value is the sum of the current reward plus a discounted Q value from the target critic network $\theta^{Q'}$ for the next control interval $t + \Delta t$. γ is called

the discount factor. Once the loss function is calculated, the parameters of the behavior critic network θ^Q are updated based on the gradient, as shown by line 13. η_Q is called the learning rate.

The loss function of the actor network is defined to maximize the Q value:

$$\max \frac{1}{M} \sum_{i=1}^M Q(s^{(i)}(t), a^{(i)}(t); \theta^Q) | a^{(i)}(t) = \pi(s^{(i)}(t); \theta^\pi) \quad (5.5)$$

In Eq.(5.5), $a^{(i)}(t)$ is generated from the actor network $\pi(s; \theta^\pi)$. Hence, the chain rule is applied in line 14 to calculate the gradient of the Q value to the θ^π . In line 16, the parameters of the target critic network and the target actor network, $\theta^{Q'}$ and $\theta^{\pi'}$, are updated at a slower rate than the behavior network, where τ is a number between 0 and 1 and close to 1. The function of this slower update is to increase the stability of the learning. The complete deep RL-based control framework of multi-zone HVAC system is shown in Figure 5.2.

Algorithm 1: DDPG method for multi-zone HVAC control

- 1: Initialize the parameters of the critic network $Q(s,a;\theta^Q)$ and the actor network $\pi(s;\theta^\pi)$
 - 2: Initialize the target networks $Q(s,a;\theta^{Q'})$ and $\pi(s;\theta^{\pi'})$ with θ^Q and θ^π
 - 3: **for** episode = 1 to arbitrary number **do**
 - 4: Initialize system state $s(T_{out}(0), T_{in,z}(0), T_{lower}(0), \lambda^{retail}(0))$
 - 5: **for** $t = 1$ to N_T **do**
 - 6: **if** $t == k\Delta t$, where k is an integer, **do**
 - 7: Select the multi-zone HVAC control action $\text{Setpt}_z(t)$ with $\pi(s;\theta^\pi)$ plus noise
 - 8: Execute $\text{Setpt}_z(t)$, receives the immediate reward $r(t)$ and the next state $s(t + \Delta t)$
 - 9: Store the transition $(s(t), \text{Setpt}_z(t), r(t), s(t + \Delta t))$ in the replay buffer
 - 10: **end if**
 - 11: Collect a mini-batch of transitions $(s^{(i)}(t), \text{Setpt}_z^{(i)}(t), r^{(i)}(t), s^{(i)}(t + \Delta t))$ with the size M from the replay buffer
 - 12: Calculate the MSE of the Q value:
$$q^{target(i)}(t) = r^{(i)}(t) + \gamma Q(s^{(i)}(t + \Delta t), \pi(s^{(i)}(t + \Delta t); \theta^{\pi'}); \theta^{Q'})$$
$$L(\theta^Q) = 1/M \sum_{i=1}^M (q^{target(i)}(t) - Q(s^{(i)}(t), \pi(s^{(i)}(t); \theta^\pi); \theta^Q))^2$$
 - 13: Update the parameters of the critic network:
$$\theta^Q = \theta^Q - \eta_Q \nabla_{\theta^Q} L(\theta^Q)$$
 - 14: Calculate the gradient of the Q value to the actor network parameter θ^π :
$$\nabla_{\theta^\pi} J \approx 1/M \sum_{i=1}^M \nabla_{\pi} Q(s^{(i)}(t), \pi(s^{(i)}(t); \theta^\pi); \theta^Q) \nabla_{\theta^\pi} \pi(s^{(i)}(t); \theta^\pi)$$
 - 15: Update the parameters of the actor network:
$$\theta^\pi = \theta^\pi - \eta_\pi \nabla J$$
 - 16: Update the parameters of the target network with a smaller step:
$$\theta^{Q'} = (1 - \tau) \theta^Q + \tau \theta^{Q'}$$
$$\theta^{\pi'} = (1 - \tau) \theta^\pi + \tau \theta^{\pi'}$$
 - 17: **end for**
 - 18: **end for**
-

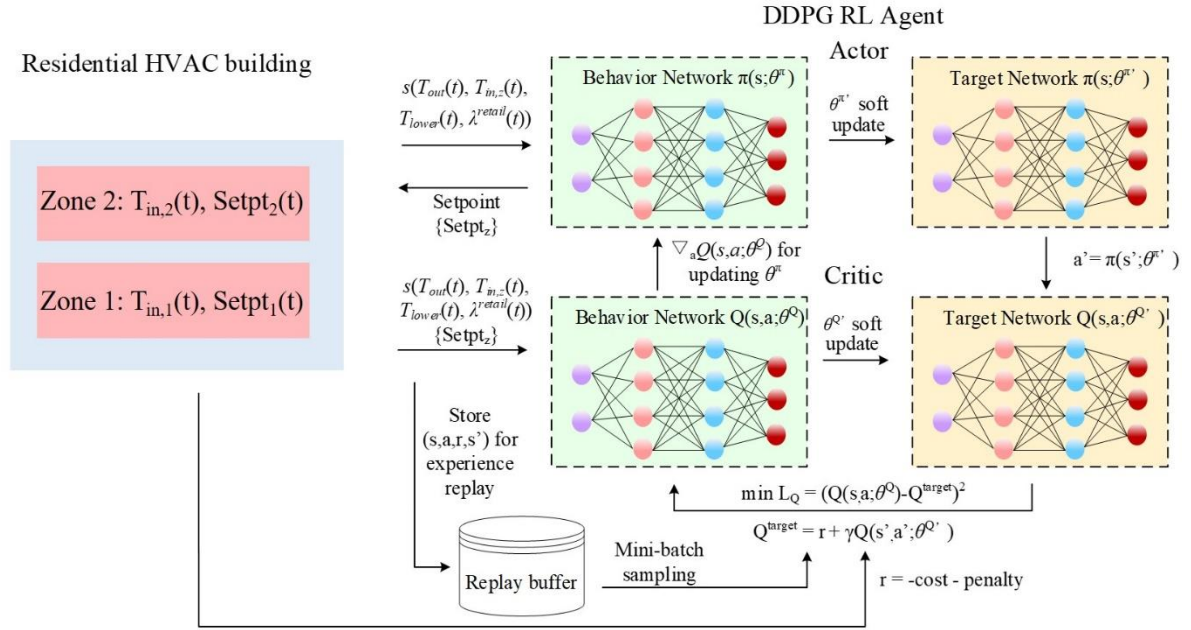


Figure 5.2. Multi-zone HVAC control framework with DDPG

5.4 Case Study

In this section, the effectiveness of the applied DDPG-based continuous control method for multi-zone residential HVAC is demonstrated through simulations with real-world data, as well as by comparison with the DQN-based discrete control method and the benchmark cases.

5.4.1 Simulation environment

A two-zone residential HVAC model [130] is implemented for training and testing the applied deep RL method, with real-world weather data during 2019-2020 obtained from [131]. For price signals, a simulated retail price sequence is generated, which include a high price value and a low price value. The price is regularly switched between the two values every three hours. The reason for applying such a frequently changing price sequence is to find if the deep RL agent can identify the effect of price signals on the reward function and to properly adjust its control strategies. It is further assumed that the lower bound of the user comfort level changes four times during the daily cycle, as shown in Table 5.1:

Table 5.1. Daily user comfort level

Time period	0:00 – 6 :00	6:00 – 12 :00	12:00 – 18 :00	18:00 – 24 :00
$T_{\text{lower}}(^{\circ}\text{C})$	18	17	18	19

The control interval of the RL agent is set to 60 minutes, i.e., $\Delta t = 60$. Since we only focus on the heating effect of the HVAC system, the November weather data is used as the training data. During the training, one episode is defined as 24 hours. In this way, 24 ($s^{(i)}(t)$, $\text{Setpt}_z^{(i)}(t)$, $r^{(i)}(t)$, $s^{(i)}(t + \Delta t)$) transitions will be generated from each episode. In total 300 episodes are simulated for the RL agent to learn. After the training, the RL agent will be applied to new test days with different weather conditions to examine its generalization and adaptability.

5.4.2 Design of the DNN structure in Deep RL

The detailed design of the actor and critic network in DDPG is shown in Table 5.2. The design of DQN is also listed for comparison. The designs of both DDPG and DQN are obtained via a trial-and-error process, and the current configurations provide the best possible results among all the trials.

Table 5.2. DNN structure applied in DDPG and DQN algorithm

Algorithm	DDPG		DQN
	critic network	actor network	
Size of input	[1,7]	[1,5]	[1,5]
No. of hidden layers	2	2	2
Size of each hidden layer	[7,20], [20,10]	[5,20],[20,10]	[5,20],[20,10]
Size of output	[1]	[2]	[25]
Activation function for the hidden layer	ReLU	ReLU	ReLU
Optimizer	Adam	Adam	Adam
Learning rate(η)	0.001	0.01	0.01
Discount factor(γ)	0.99	-	0.99
Batch size	48		
Weights of the reward	$\omega_c : 10, \omega_p : 1$		

For the DDPG method, the input to the critic network is a vector containing both state variables and action variables, and the output is the estimated Q value, which is a scalar; the input to the actor network is a vector containing only state variables, and the output is a vector containing the setpoint for each zone. Although the setpoint is a continuous variable, in reality there is always a range of the setpoint for maintaining user comfort. Therefore, the output layer from the actor network utilizes tanh as the activation function, which confines the output with a range of $[-1,1]$. The actual setpoint is calculated as $\text{Setpt}_z = T_{\text{lower}} + \Delta T \cdot (y_{\text{out}} + 1)$, where y_{out} is the output from the actor network, and ΔT is the upper range of the setpoint. In the simulation, ΔT is set to 2°C . Therefore, the setpoint selected by the DDPG lies within the range of $[T_{\text{lower}}, T_{\text{lower}} + 2]$.

For the DQN method, the input is also the state variables. Since DQN requires a discrete action space, we discretize the range of setpoint with a step size of 0.5°C . As a result, there are 5 actions for each zone and 25 combinations of actions for the 2-zone HVAC. The output from DQN is a vector containing 25 Q values, with each corresponding to one combination of actions.

5.4.3 Performance of the continuous HVAC control method

1) Convergence of the DDPG

In Figure 5.3, the average returns gained after each episode during the training process in the DDPG and DQN are presented. Notice that the average return in the first few episodes appears to be higher than that of the last few episodes. This is because for each episode, one training day is randomly chosen. Some training days may have moderate outdoor temperature, which can lead to lower energy cost and lower penalty, and vice versa. However, as the training proceeds, the number of episodes grows, and the average return is neutralized. Both curves gradually become steady as the training evolves. However, the average return gained by DDPG method is higher than that of the DQN method. This is because the size of the output from DQN is larger than that of DDPG, and the combination of actions have not been fully explored after 300 episodes, leading to a lower average return.

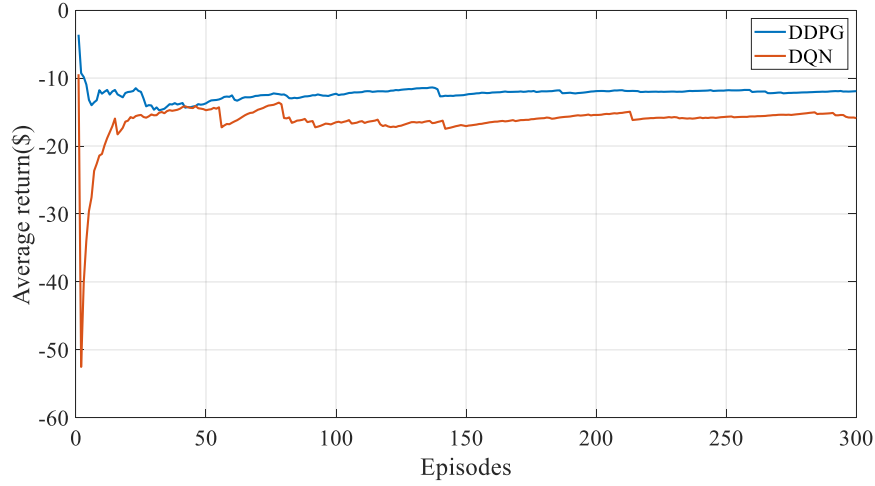


Figure 5.3. Convergence of different deep RL methods

2) Computational efficiency

After the training process, the DDPG RL agent is applied to 10 test days in January 2020 from the real-world data in [132] to generate the optimal HVAC control strategy. The time cost is around 19 seconds for testing, which is highly time-efficient. The code is written in Python 3.6 with the open-source deep learning platform TensorFlow [133]. The hardware environment is a laptop with Intel®Core™ i7-7600U 2.8 GHz CPU, and 16.00 GM RAM.

3) Comparison of DDPG with DQN and the benchmark cases

In this study, the well-trained deep RL agents from both DDPG and DQN are run on new test days to verify their learning performance. We also design two benchmark cases without the RL agent as comparisons. The benchmark cases are described as follows: a) Rule-based case: the setpoint is set at the lowest value at the peak price hours, and the highest value at the off-peak price hours, to realize the pre-heating effect to save energy cost; b) Fixed setpoint case: the setpoint is always at the highest value of the setpoint range to avoid any temperature violation.

The final optimized results of the RL methods and the benchmark cases are shown in Table 5.3:

Table 5.3. Test results of different HVAC control methods

Control method	DDPG	DQN	Rule-based	Fixed setpoint
Total cost (\$)	55.21	65.03	39.08	71.48
Temperature violation (minutes)	48	230	2617	0
Average temperature violation (°C)	0.13	0.93	1.85	0

In Table 5.3, the well-trained deep RL agents are applied to generate the HVAC control strategies for the first 10 days in January 2020. The weather conditions of the test days are different from that of the training days, since the outdoor temperature is much lower in January than in November. The total cost in the table refers to the total energy cost over the 10 days, and the temperature violation in the table refers to the total number of minutes that the indoor temperature falls below $T_{\text{lower}} - T_{\text{th}}$, as shown by Eq. (3). T_{th} is set to 0.3 °C. The average temperature violation indicates on average by how many degrees the indoor temperature is lower than the setpoint. As shown in the table, the control strategy derived from DDPG method has both lower energy cost and fewer temperature violations than that of the DQN. With regard to the benchmark cases, in the rule-based case, because the pre-heating logic is applied based on the price structure, it obtained the lowest cost among all four cases. However, by always setting the setpoint to the lowest value at peak price hours, this control strategy results in severe temperature violation. In the fixed setpoint case, since the setpoint is always set at the highest value, there is no temperature violation. However, the energy cost is also the highest among the four cases. The control strategy and the indoor temperature in the four cases are further illustrated in Figure 5.4.-Figure 5.6.

In all the figures, the yellow rectangular area represents the feasible region of the setpoint [T_{lower} , $T_{\text{lower}} + 2$ °C]. As can be observed, the setpoint range changes at a daily cycle. In addition, the indoor temperature in zone 1 is lower than that of zone 2, this is because in the building model, zone 1 is on the 1st floor and zone 2 is on the 2nd floor, and the warmer air goes to upper floors.

In Figure 5.4., the DDPG RL agent develops a setpoint control strategy that when the outdoor temperature is relatively high, i.e. in the first 4,000 minutes, the setpoint will be set at the lowest

value at the peak price hour, and at the highest value at the off-peak hour, to realize the pre-heating effect and to reduce energy cost, which is similar to the control logic of the rule-based case. When the outdoor temperature is low, i.e., in the last 2,000 minutes, the setpoint is always set at the highest value to avoid the indoor temperature violation. On the contrary, in the rule-based case, the control strategy still follows the price structure even when the outdoor temperature is extremely low, which results in severe indoor temperature violation, as shown in Figure 5.6. Such comparisons indicate that after the training, the DDPG RL agent has acquired the knowledge that the price signal and the outdoor temperature has a significant impact on the reward, and it learns to intelligently set the setpoint based on this state information to reach a higher reward value.

The control strategy of DQN RL agent is shown in Figure 5.5. When the outdoor temperature is relatively high, i.e. in the first 4,000 minutes, the setpoint is set at a relatively high value, and it does not follow the change of retail price. When the outdoor temperature is extremely low, i.e., around 12,000 minutes, the setpoint is set at the lower bound, which results in temperature violation. The DQN RL agent has not successfully capture the impacts of the state variables on the reward function. This can be attributed to the large number of action combinations encountered by the Q network. In such a case, the DQN RL agent has not fully explored all of the possible action combinations to maximize the reward, thus obtains a control strategy with higher energy cost and more temperature violations.

Finally, in Figure 5.7, the fixed setpoint case, since the setpoint is always set at the highest value, the indoor temperature for both zones also remain at the highest level among the four test cases. However, this fixed setpoint case results in the highest energy cost.

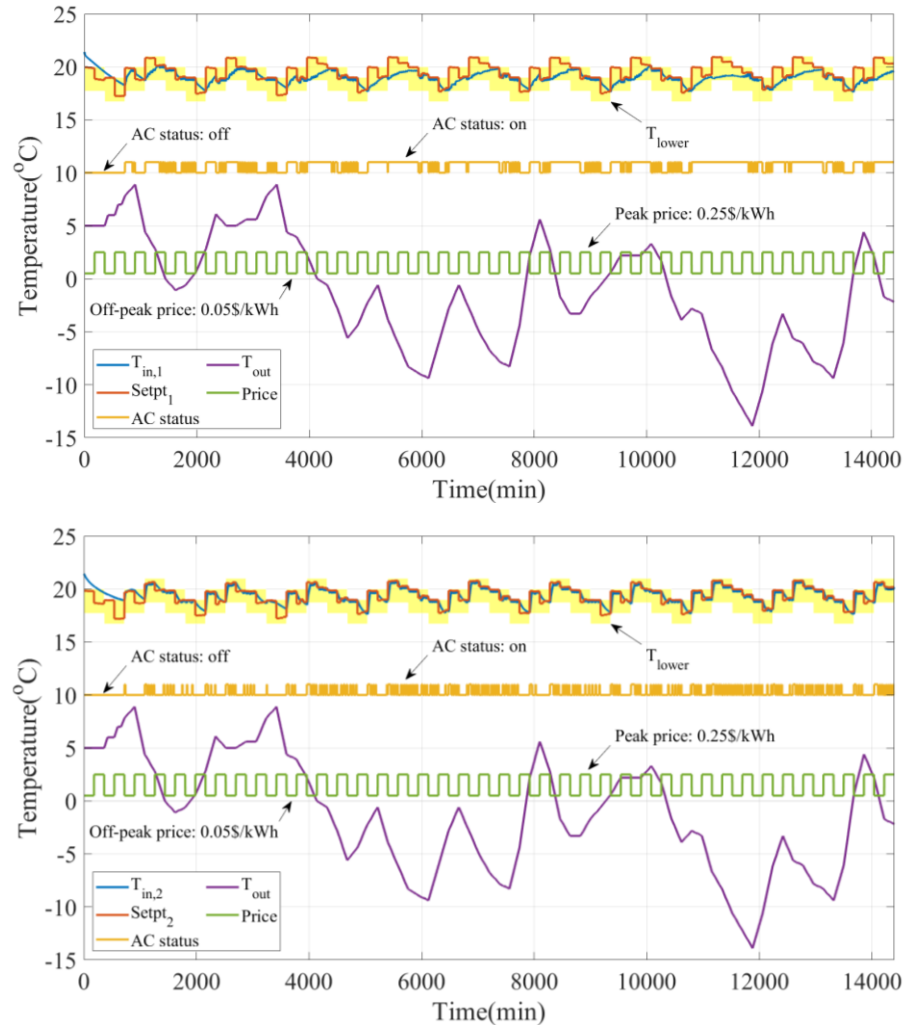


Figure 5.4. Setpoint control strategy based on DDPG for 10 test days (top: zone 1; bottom: zone2)

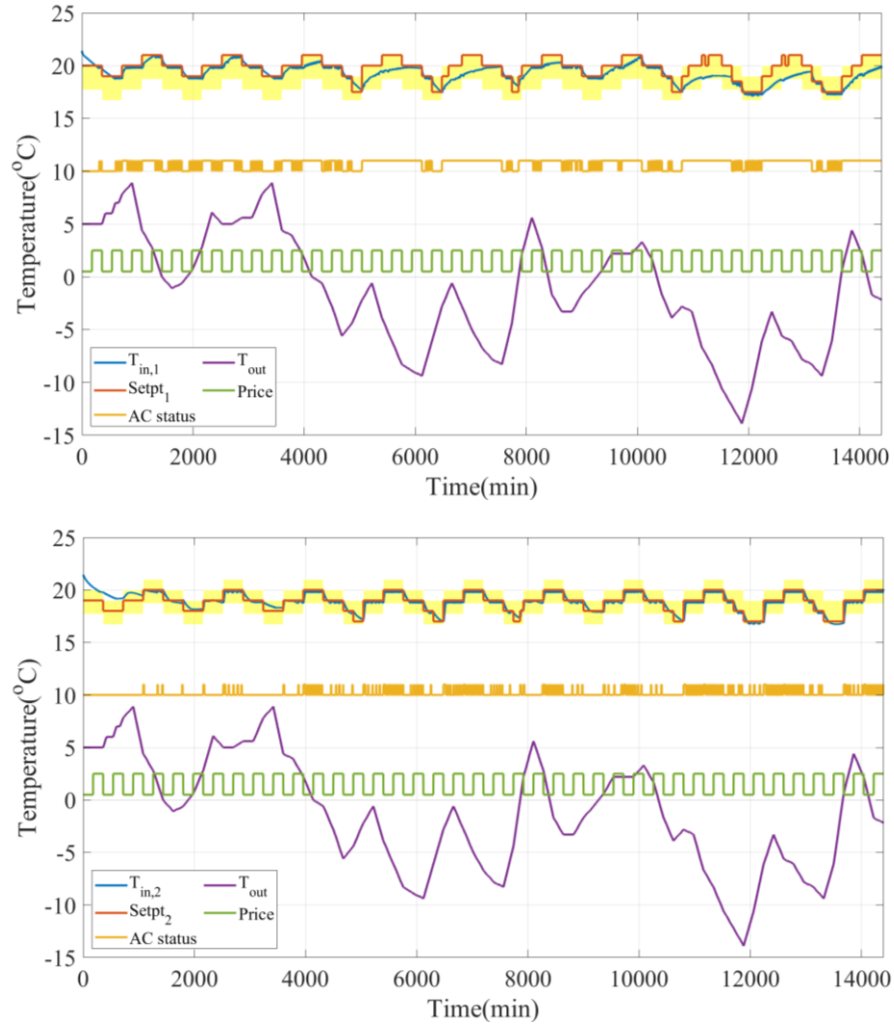


Figure 5.5. Setpoint control strategy based on DQN for 10 test days (top: zone 1; bottom: zone2)

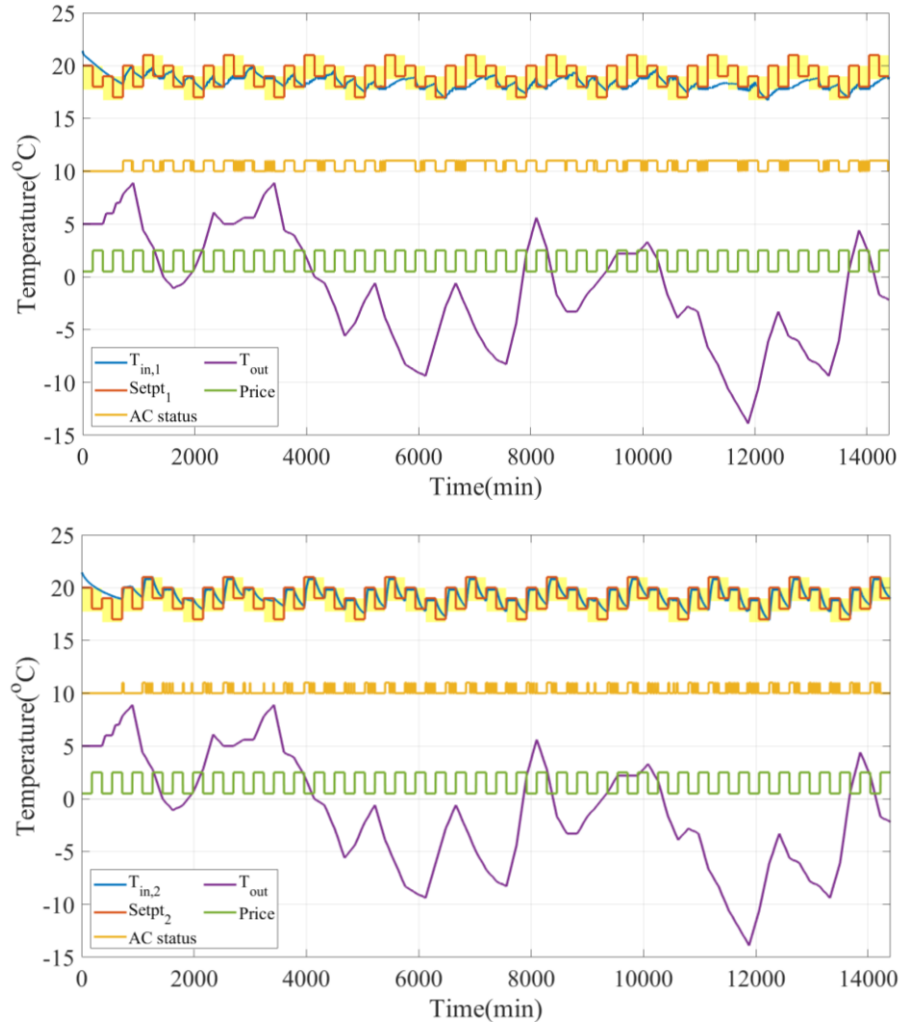


Figure 5.6. Setpoint control strategy from the rule-based case for 10 test days (top: zone 1; bottom: zone2)

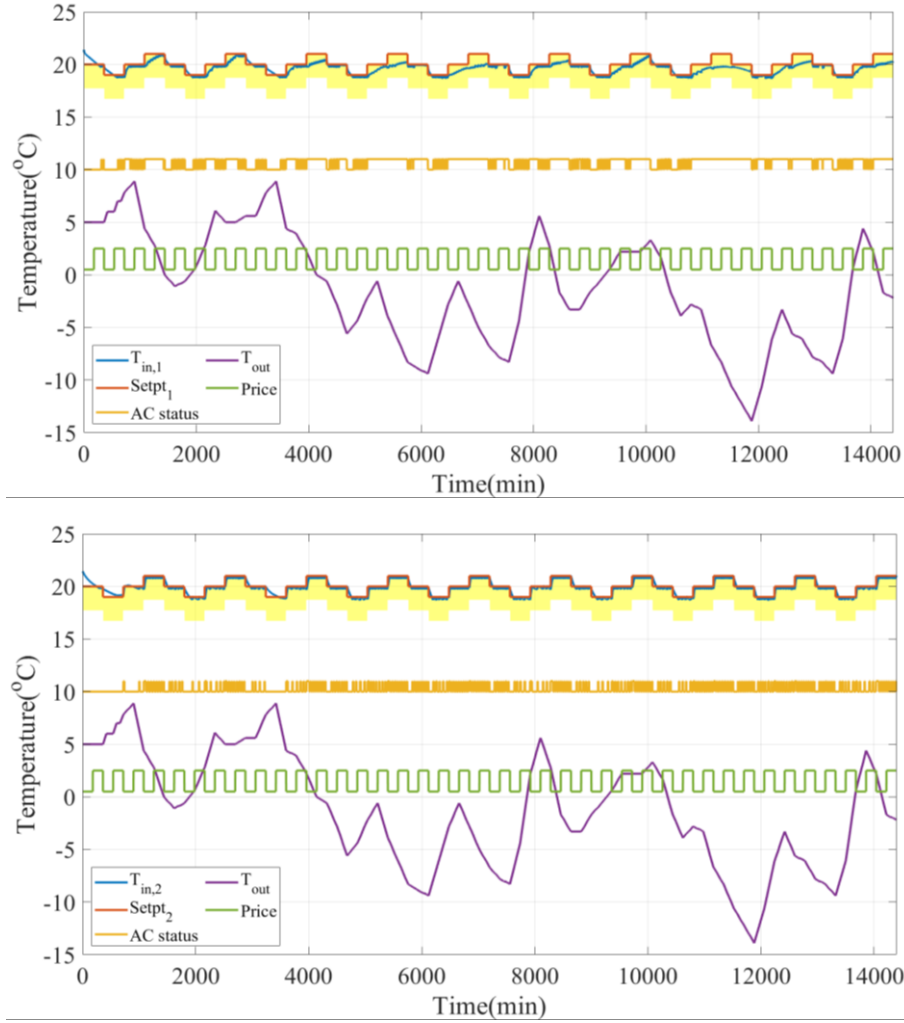


Figure 5.7. Setpoint control strategy from the fixed setpoint case for 10 test days (top: zone 1; bottom: zone2)

4) Generalization of DDPG Algorithm

a) Extending DDPG RL agent to different residential buildings

The well-trained DDPG RL agent is further tested in new residential building models with HVAC system to fully validate its generalization and robustness. Ten building models are generated with different thermal mass parameters, the variation of which follows a normal distribution. The same 10 test days in January 2020 are applied in this case. The energy cost and the temperature violation for the 10 building models under the DDPG control strategy and under

the two benchmark cases are compared in Table 5.4 and Figure 5.8. . As shown in the table, similar to the results in Table 5.3, the rule-based control strategy provides the lowest energy cost, while the fixed setpoint control strategy provides the lowest violation. The well-trained DDPG RL agent can obtain an HVAC control strategy that properly weigh the two objectives, resulting in a relatively lower energy cost and fewer temperature violations for different test building models. Therefore, it can be safely concluded that the DDPG RL agent can flexibly adapt to unseen physical environments and provides an economic HVAC control strategy after its offline training within the fixed environment.

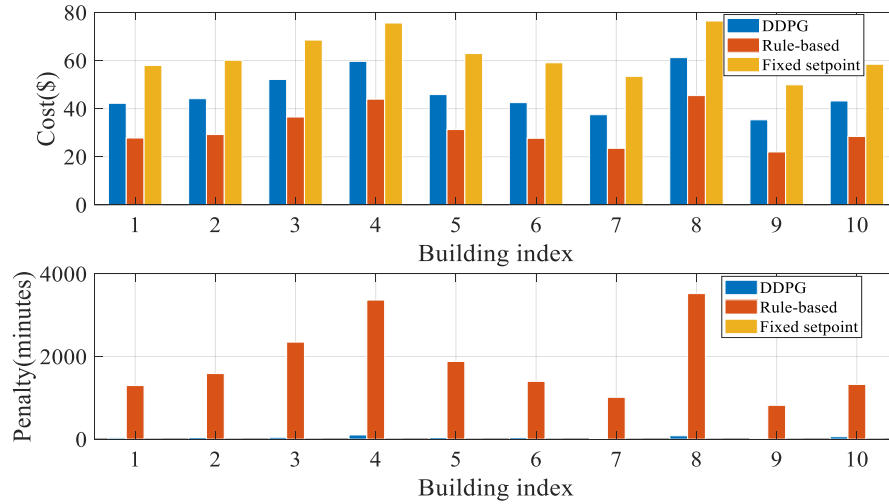


Figure 5.8. Illustration of the comparison results

Table 5.4. Comparison of optimization results for different building models

Building index	DDPG		Rule-based		Fixed setpoint	
	Cost (\$)	Temperature violation (minutes)	Cost (\$)	Temperature violation (minutes)	Cost (\$)	Temperature violation (minutes)
1	42.22	31	27.78	1296	57.98	0
2	44.13	41	29.22	1586	60.13	0
3	52.14	45	36.51	2347	68.52	0
4	59.66	101	43.94	3364	75.61	0
5	45.84	41	31.30	1879	62.91	0
6	42.49	39	27.68	1398	59.06	0
7	37.47	24	23.51	1012	53.42	0
8	61.21	81	45.42	3520	76.44	0
9	35.34	25	21.98	818	49.90	0
10	43.19	59	28.41	1323	58.46	0

b) DDPG performance under different retail price signals

In the above simulations, a simulated retail price sequence is generated for training and testing the deep RL agent, which is simply composed of only two price signals. To demonstrate that the well-trained DDPG RL agent has developed high generalization to an unseen environment without additional training, the DDPG RL agent is further tested with the PJM wholesale real-time hourly locational marginal price (LMP) data [133]. The retail price is set at 3 times of the wholesale market price. The PJM price changes hourly and fluctuates within a large range. The final optimized results of the two deep RL methods and the benchmark case are shown in Table 5.5:

Table 5.5. Test results of different control methods (under PJM price)

Control method	DDPG	DQN	Fixed setpoint
Total cost (\$)	32.90	31.80	32.71
Temperature violation (minutes)	0	222	31
Average temperature violation (°C)	0	1.00	0.27

In Table 5.5, the fixed setpoint case applies a control strategy where the setpoint is always set at the middle of the setpoint range. This is because the PJM price sequence contains more than just two values, and it cannot be simply divided into two groups as high price and low price. As a result, the setpoint is set at the middle point to avoid possible temperature violations while minimizing the energy cost.

The control strategy and the associated indoor temperature in the three cases are further illustrated by Figure 5.9-Figure 5.11. As can be observed in the figure, the PJM price demonstrates a very different pattern from the simulated price sequence. For most of the time the price remains at a relatively low level, with some occasional spikes and fluctuations. However, the well-trained DDPG RL agent still attempts to follow the price tendency, and intelligently sets the setpoint to realize the pre-heating effect. For example, a price spike appears around 12,500 minutes, the DDPG RL agent catches this sudden change, and lowers the setpoint. Around 13,500 minutes the retail price sequence demonstrates some fluctuations, and the DDPG RL agent also adjusts the setpoint accordingly. It should be pointed out that under the price signals that are more time-variant like the PJM market price, it is difficult to develop a simple rule-based control strategy, because the price range is uncertain. However, the well-trained DDPG RL agent can still work intelligently under such uncertain environment, and obtain satisfying economic benefits. Therefore, the adaptability of the DDPG algorithm is proved, which makes it feasible for real-world online applications.

In Figure 5.10, the HVAC control strategy developed by the DQN RL agent also intends to follow the retail price tendency. However, at the price pike period (12,500 minutes) and the price variation period (13,500 minutes), the DQN RL agent chooses the lowest setpoint values, which results in temperature violation in zone 1, as shown in the bottom figure.

Finally, in Figure 5.11, the fixed setpoint case also leads to some temperature violations in zone 1 when the outdoor temperature is extremely low (after 10,000 minutes).

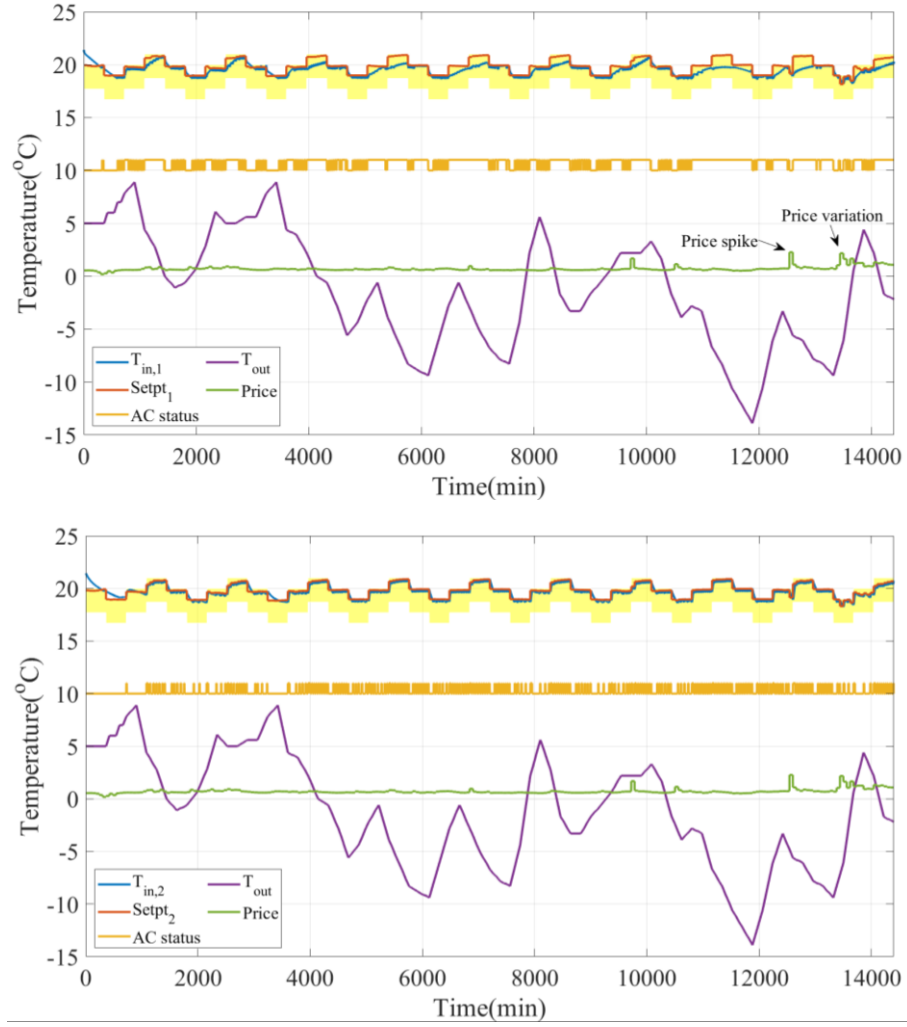


Figure 5.9. Setpoint control strategy based on DDPG under PJM price for 10 test days (top: zone 1; bottom: zone2)

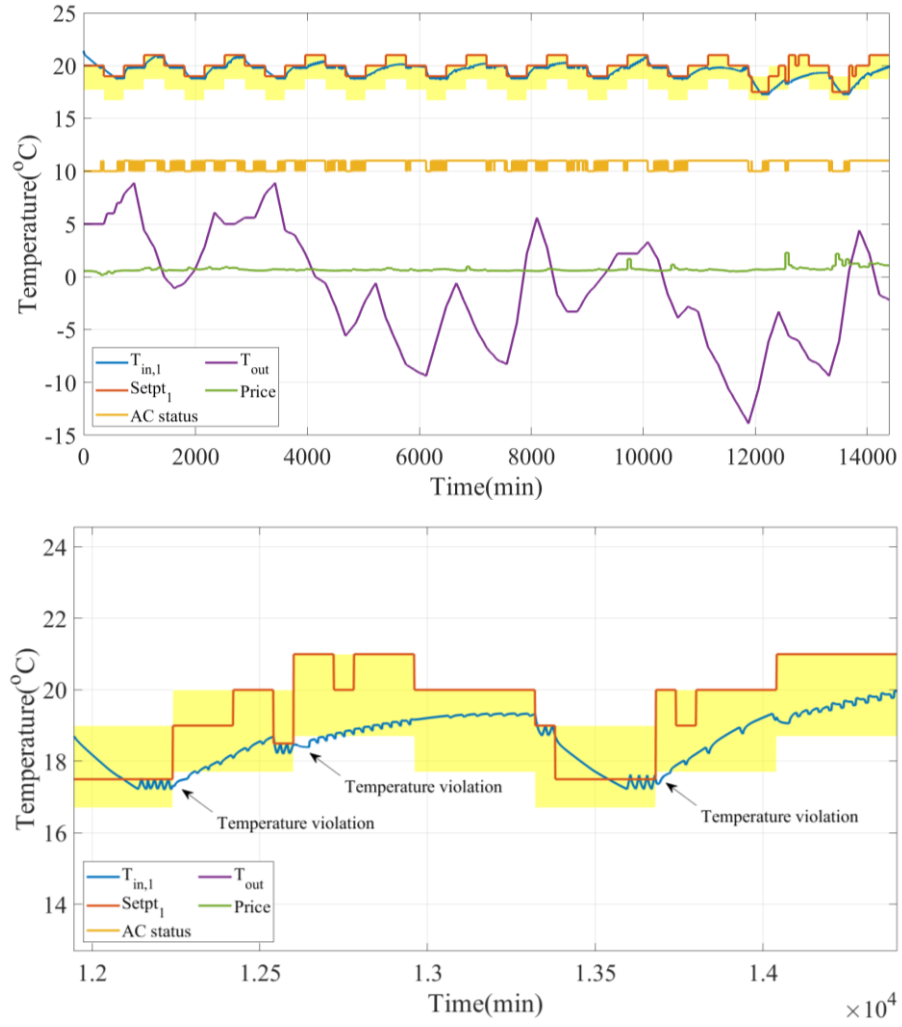


Figure 5.10. Setpoint control strategy based on DQN under PJM price for 10 test days (top: zone 1; bottom: zoomed part of zone 1)

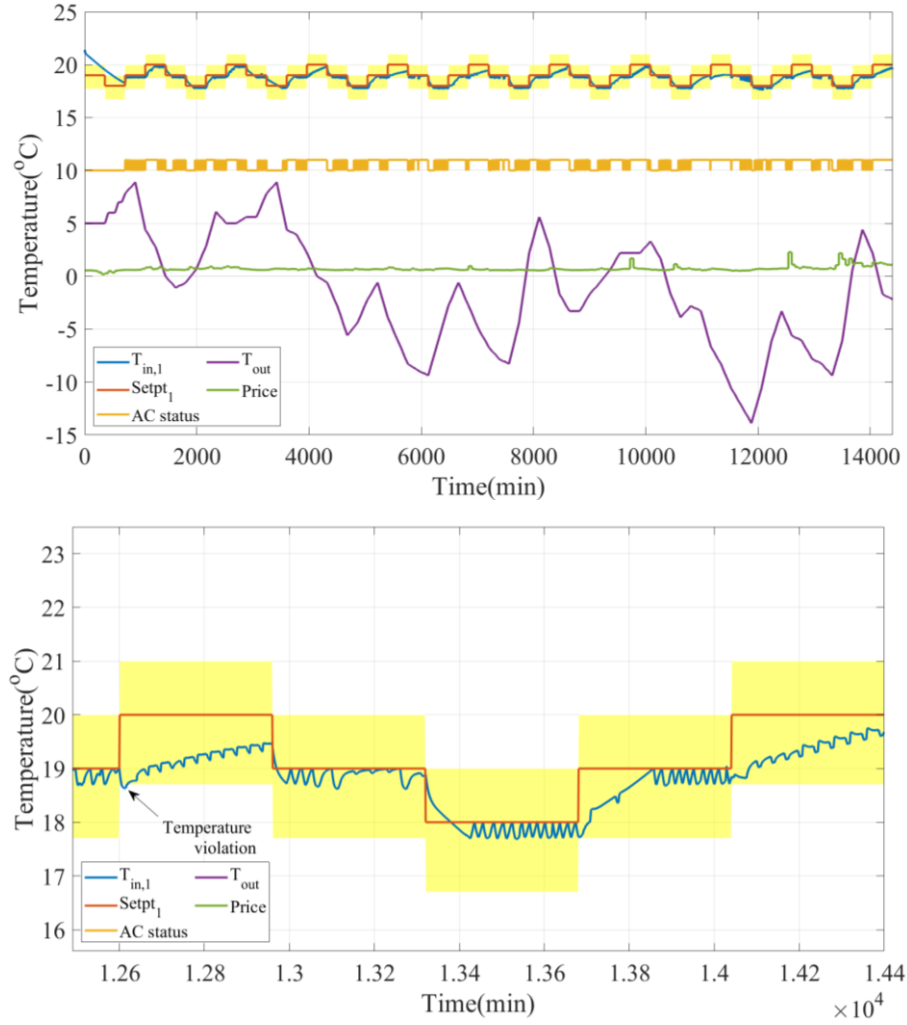


Figure 5.11. Setpoint control strategy in the fixed setpoint case under PJM price for 10 test days (top: zone 1; bottom: zoomed part of zone 1)

5.5 Conclusions

In this paper, a deep RL method, the DDPG is applied for controlling the multi-zone residential HVAC system to minimize the energy consumption cost while maintaining the user comfort. The DDPG can realize a continuous control of the HVAC setpoint due to its application of the DNNs. Simulation results demonstrate that the well-trained DDPG RL agent is able to adapt to the unknown environment and intelligently decides its setpoint control strategy to maximize the economic benefit. The training efficiency and high generalization ability of the DDPG algorithm

makes it a potential tool for future online applications in solving MDP problems with hidden information or with continuous search space.

For future works, one interesting direction is to let the deep RL agent automatically switch between different operation modes, i.e. cooling and heating, to adapt to different seasoning scenarios. In this way, once the deep RL is well-trained, it can be directly applied to a longer control period, i.e. one year, to provide economic control strategies for HVAC users. Another exploratory direction is to let the deep RL agent learn a more variant setpoint schedule customized by users, given that different HVAC users can have different comfort level. By investigating these two directions, the deep RL agent will become more generalized and robust against uncertainties in real-world operation scenarios.

5.6 Acknowledgement

This work is funded by the Department of Energy, Energy Efficiency and Renewable Energy Office under the Buildings Technologies Program.

Chapter 6 Solving Markov Game in Day-ahead Electricity Market with Multi-Agent Deep RL

In this chapter, the day-ahead electricity market bidding problem with multiple strategic GENCO bidders is studied. The problem is formulated as a Markov game model, where GENCO bidders interact with each other to develop their optimal day-ahead bidding strategies. Given the uncertainties and unobservable information embedded within the problem, a model-free, data-driven approach for solving the above Markov game is presented in this paper based on the latest DL technique called multi-agent deep deterministic policy gradient (MADDPG) algorithm. The MADDPG algorithm is a type of deep RL method designed for continuous control within a multi-agent competitive environment, where DNNs are combined with reinforcement learning to capture the intricate environmental variations and to optimize decisions. The applied MADDPG method is implemented on the IEEE 30-bus system, with three RL agents to prove its feasibility and the computational efficiency in solving the multi-agent decision-making problem. The obtained bidding strategies for each GENCO bidder are further examined from the viewpoint of executing market power to provide insights for market participants.

6.1 Introduction

In power systems, the strategic market bidding problem of generation companies (GENCOs) is a complex real-world multi-stage decision optimization process. The topic remains interesting ever since the deregulation of electricity market. The problem is investigable because there are many unknowns in the electricity market, e.g. the offers of other bidders. Also, the problem of executing market power is intriguing, where market bidders with large capacity can manipulate the market clearing results for his own benefits.

In the literature, many research efforts are dedicated to cracking the problem of strategic market bidding. The non-cooperative game model is initially introduced in [134],[135] to solve the

problem of pricing electricity in a deregulated energy marketplace, where the unknown information of opponent bidders' costs is equivalently represented by probabilities, and a game with complete but imperfect information can be formed and solved analytically to obtain Nash Equilibrium (NE). The authors in [136] further consider the probability of forming coalitions among generators under the non-cooperative game framework, where coalition members coordinate their bidding strategies to maximize the common profit. In ref. [137], an iterative simulation method is presented to obtain the desired NE, where each GENCO constantly updates their bidding curves based on the latest market clearing results until there is no increase in their gains. To fully capture the uncertainty of the rivals' bids, robust optimization is implemented in [138], which formulates the bidding problem as a bi-level max-min model to obtain the feasible region of the desired bidding strategies.

All of the above works can be categorized as model-based methods, where the uncertainties of the electricity market are overcome either by forecasted model or via iterative interactions. The model-based method can provide analytical insights into market equilibrium. However, achieving convergence with model-based methods generally requires strict prerequisites, e.g., linear or convex objective functions. In addition, the real-world market structure is not normally designed for iterative negotiations among the participants, which can be both time-consuming and resource-consuming.

The data-driven RL method stands out as an efficient alternative to the conventional model-based method. Contrary to the latter, the RL method does not assume the knowledge of the exact mathematical model, but it approximates the model through continuous interactions with the actual environment. The RL method is well known for handling multi-stage decision-making problems, where game theory may fail due to its limitations in solving time-dependent models. The RL method can also adapt to new changes in the environment, where the solutions to the model-based method are fixed to the pre-defined model.

The RL algorithm for the GENCO optimal bidding problem is first introduced in [139], where

the day-ahead market bidding for the next 24 hours is formulated as a Markov decision process (MDP), and an actor-critic RL method is applied for GENCO decision making. The actor is used for selecting an action from an existing bidding policy, and the critic is used to evaluate the selected action based on temporal difference error. The Q-learning method is applied in [140] in a repeated Cournot game to maximize the accumulative payoff of a GENCO in day-ahead market bidding. The learned bid proves to converge near the desired Nash-Cournot equilibrium point. Authors in [141] propose the fuzzy Q-learning method to handle the variations of renewable power penetration in hour-ahead market bidding, which demonstrates higher computational efficiency than the conventional Q-learning method. The authors in [142] discuss the effects of risk aversion on Q-learning performance and the bidding strategies of GENCOs. In [143], a multi-agent Q learning method is designed to achieve bidding strategies for multiple competitive generators.

Popularized by the AI computer program AlphaGo, the deep RL represents the latest development for the data-driven RL method for solving time-sequential decision-making problem with partial or hidden information. Deep RL is a combination of a DNN and RL. The highlight of the deep RL method, compared to other learning-based methods discussed above, is that it can build a generalized model with DNN that can adapt to continuous environmental variations. The success of deep RL has been witnessed in the fields of computer games, robotics and industrial automation, etc. In power systems, the potentials of implementing deep RL for demand-side energy management and electric vehicle charging/discharging scheduling are shown in [40],[43]. The deep deterministic policy gradient (DDPG) algorithm is applied to solve the bidding problem of a load serving entity and a single producer in [44], [144]. The authors in [145] further explore the market equilibrium with multiple strategic GENCO bidders using both deep policy gradient method and long-short-term (LSTM) memory neural network.

Motivated by the above researches, in this chapter we also focus on jointly optimizing the bidding strategies of multiple GENCOs in the day-ahead market with multi-agent deep RL method. The main contributions of the chapter are summarized as follows:

- A Markov game is formulated to describe the strategic day-ahead market bidding process of multiple GENCOs as price makers. Under such situation, each GENCO acts intelligently to maximize its own benefits with the consideration of bidding policies from other rivals.
- A novel MADDPG method is applied to solve the above Markov game. Compared with the existing approaches, the method implements a centralized training and decentralized execution mechanism, which can deal with non-stationary environments where the rational players constantly change their strategies, as well as high-dimensional continuous state and action spaces.
- A comprehensive simulation analysis is presented to prove that the applied MADDPG method can achieve the optimal bidding strategy with the absence of market power, and it also has high generalization to perform intelligently under unseen market environment. The superiority of the method is further verified by comparing with baseline cases.

The rest of the chapter is organized as follows: section 6.2 provides the formulation of the multi-agent day-ahead market bidding problem under the context of Markov game; section 6.3 introduces the MADDPG method for multi-agent decision-making process within non-stationary environment; the simulation results and analysis are shown in section 6.4; finally, section 6.5 concludes the chapter.

6.2 Multi-agent Market Bidding Problem Formulation

6.2.1 A brief on electricity market and bidding strategies

A deregulated electricity market is usually composed of two stages, a day-ahead market and a real-time market. In the day-ahead market, GENCOs submit the amount of energy they are willing to sell for the next 24 hours, and the associated offer price, and consumers submit the amount of energy they are willing to buy and the associated bid price. The market operator clears the market by running an optimal power flow (OPF) calculation and releases the market clearing prices and quantities to the supply side and the demand side.

GENCOs with large capacities can execute market power through the following two ways:

physical withholding, which means that they submit generation quantities that are less than their capacity; or economic withholding, which means that they offer prices that are higher than their marginal cost. However, executing market power can also be risky since GENCO bidders have incomplete information about their rivals.

There are two market settlement methods: 1) the marginal price principle, where all suppliers receive the same market clearing price which is the cost of the marginal bidding block; and 2) the pay-as-bid principle, where each winning supplier receive a price based on their respective bidding prices, and it can be different from one to another. The marginal price principle is applied in almost all the organized wholesale markets in the United States, while the pay-as-bid principle is mostly seen in European countries like France and Britain.

In terms of auction theory, pay-as-bid principle is a variation of the sealed first-price auction and marginal price principle is a variation of the sealed second-price auction. It has been proved that in the sealed second-price auction, truthful bidding, which means no economic withholding, is a dominant strategy. While in the sealed first-price auction, truthful bidding is a necessary but not sufficient condition for reaching NE.

In this study, we assume that the day-ahead electricity market is cleared based on the marginal price principle, and GENCOs are allowed to execute economic withholding to maximize their profits. Under such condition, a NE bidding strategy of the GENCO g should satisfy the following two conditions:

$$(i) \max_{g' \neq g} v_{g'} \geq b_g; (ii) \max_{g' \neq g} b_{g'} \geq v_g$$

In the above two conditions, v_g is GENCO g 's valuation for the sold generation, b_g is the bidding price of GENCO g . The profit of GENCO g will be $b_g - v_g$. Condition (i) means that GENCO g bids at a sufficiently low price in order to win the bid, and condition (ii) means that the valuation of GENCO g is sufficiently low. In section 6.4, we will demonstrate that the applied deep RL method is able to achieve a NE strategy that corresponds with the above two conditions through truthful bidding.

6.2.2 Mathematical formulation of day-ahead electricity market clearing

The day-ahead electricity market clearing model is shown as the following DCOPF model:

$$\min \sum_{t=1}^{N_T} \sum_{g=1}^N \sum_{b=1}^{N_b} \lambda_{g,b}^{bid}(t) P_{g,b}^{cleared}(t) \quad (6.1)$$

$$s.t. \sum_{g \in i} \sum_{b=1}^{N_b} P_{g,b}^{cleared}(t) - \sum_{l \in i} P_l^{load}(t) = \sum_{j \in n(i)} \frac{\theta_i(t) - \theta_j(t)}{x_{ij}} \quad (6.2)$$

$$\left| \frac{\theta_i(t) - \theta_j(t)}{x_{ij}} \right| \leq F_{ij,max}, \forall i, j \in N_F \quad (6.3)$$

$$0 \leq P_{g,b}^{cleared}(t) \leq P_{g,b}^{bid}(t), \forall g, \forall b \quad (6.4)$$

In Eq. (6.1), N_T is the number of time intervals, N_b is the number of bidding blocks submitted by the g^{th} GENCO, $\lambda_{g,b}^{bid}(t)$ and $P_{g,b}^{cleared}(t)$ are the bidding price and the cleared quantity of the b^{th} bidding block. Eq. (6.2) is the power balance constraint at the i^{th} bus, where $P_l^{load}(t)$ is the load at bus i . θ_i and θ_j are the head and tail bus voltage angles at the transmission line ij . $n(i)$ is the set of buses that are connected to bus i . x_{ij} is the line reactance. Eq. (6.3) is the transmission line capacity constraint, where N_F is the set of transmission lines. Eq. (6.4) ensure that the cleared quantity does not exceed the bidding quantity submitted by the GENCO bidders.

In the DCOPF model (6.1)-(6.4), $\lambda_{g,b}^{bid}(t)$ and $P_{g,b}^{bid}(t)$ are known values submitted by GENCO bidders. They should satisfy the following constraints:

$$P_g^{min} \leq \sum_{b=1}^{N_b} P_{g,b}^{bid}(t) \leq P_g^{max} \quad (6.5)$$

$$\lambda_{g,b}^{bid}(t) = \varepsilon_g(t) \lambda_{g,b}^{cost}(t) \quad (6.6)$$

Eq. (6.5) is the capacity limit of the bidding block, where P_g^{min} and P_g^{max} are the lower and upper limit of the generation. Eq. (6.6) indicates the economic withholding of the GENCO bidder, where $\lambda_{g,b}^{cost}(t)$ is the marginal cost of the b^{th} bidding block. $\varepsilon_g(t)$ is a bidding factor between 1 and an upper limit $\varepsilon_{g,max}$, which indicates that the GENCO bidder can deliberately submit a higher marginal cost

to increase its profit.

From the above DCOPF model, the cleared $P_{g,b}^{cleared}(t)$ and cleared price $\lambda_g^{cleared}(t)$ for each GENCO bidder can be obtained. The $\lambda_g^{cleared}(t)$ is the locational marginal price (LMP). The GENCO bidder then calculates its profit according to Eq. (6.7).

$$\sum_{t=1}^{N_T} \left(\sum_{b=1}^{N_b} P_{g,b}^{cleared}(t) \cdot \lambda_g^{cleared}(t) - C_g \left(\sum_{b=1}^{N_b} P_{g,b}^{cleared}(t) \right) \right) \quad (6.8)$$

$$C_g \left(\sum_{b=1}^{N_b} P_{g,b}^{cleared}(t) \right) = \sum_{b=1}^{N_b} \lambda_{g,b}^{cost} P_{g,b}^{cleared}(t) \quad (6.9)$$

In Eq.(6.8), the first item is the income of selling power at the day-ahead market; the second item is the generation cost, which has a linear expression as shown in Eq. (6.9).

From the above mathematical models, it can be observed that the decision variable for the GENCO bidders is the bidding factor $\varepsilon_g(t)$, and the decision variable for the market operator is the cleared quantity $P_{g,b}^{cleared}(t)$. Deciding the optimal bidding factor is an involved task for GENCO bidders because the bidding information of their rivals remains unknown. The conventional model-based method can fail due to this unobservability. In the following sections, the multi-agent day-ahead market bidding problem will be transformed into a Markov game, and a model-free deep RL method will be introduced as a solution.

6.2.3 Markov game model of day-ahead electricity market bidding

Before building the Markov game model, we propose the following assumptions regarding the day-ahead market bidding problem [146]:

- 1) The GENCOs submit hourly bidding blocks for the next 24 hours in the day-ahead market. The bidding quantities are their true generation capacities, only the bidding price is allowed to change.
- 2) The bidding price for the same bidding block is allowed to vary from hour to hour. However, the ratio of the highest bidding price to the lowest bidding price for the same bidding block should not exceed a threshold th_I .

3) For any two consecutive hours, the ratio of the bidding prices for the same bidding block should not exceed a threshold th_2 .

From assumption 3), it can be discovered that the bidding price for the current hour is related to the bidding price in the previous hour, which leads to a finite MDP with discrete time steps.

When multiple agents are considered in the day-ahead market bidding, the MDP is extended to a partially observable Markov game. A Markov game for N agents consists of a set of states $\mathbf{s}$, a set of observations made by each agent at the current state, $o_1, o_2, \dots, o_N$, and a set of actions $a_1, a_2, \dots, a_N$ taken by each agent based on their respective observations. After the execution of the actions, the environment will transfer to the next state following a transition probability p : $\mathbf{s} \times a_1 \times a_2 \times \dots \times a_N \times \mathbf{s} \rightarrow [0, 1]$. Each agent will receive a reward $r_i: \mathbf{s} \times a_i \rightarrow \mathbb{R}$ and a private observation for the next state $o_i: \mathbf{s} \rightarrow o_i$. The objective of each agent is to maximize the total discounted reward for the finite time steps: $R_i = \sum_{t=1}^{N_r} \gamma^{t-1} r_{i,t}$, where γ is a discount factor to convert future rewards to the present value.

In the day-ahead market bidding problem, under the context of a Markov game, the agent is each independent GENCO bidder; the private observation for each GENCO is the demand quantity for the current hour, and its bidding price at the previous hour; the state is simply defined as the summation of the observations of all GENCOs; the action is the bidding price for the current hour; and the reward is the hourly profit. The day-ahead market bidding process is a sequential decision-making problem with multiple decision makers involved, which requires that each GENCO bidder be farsighted enough to consider potential future outcomes in order to maximize the total profit.

Note that in the general day-ahead market bidding, the GENCO bidders are required to submit their bidding blocks for the next 24 hours in one shot; while in the above Markov game, the bidding decision process is decomposed to discrete time steps and the bidding price for each time step is decided sequentially. This decomposition is acceptable because at each time step, the private observation only includes the current hourly load and the bidding price at the previous hour, and does not involve any market clearing results. Hence, after the applied deep RL algorithm is well-

trained for solving the above Markov game model, it will only need the load data for the next 24 hours as input and can generate the bidding prices for the next day in one shot (given an initial bidding price) during the test process. Therefore, the algorithm can be physically implemented without violating market rules.

6.3 MADDPG Method for Day-ahead Electricity Market Bidding

6.3.1 An overview of reinforcement learning method

The RL method aims to solve the MDP process with the objective of maximizing the total discounted reward $R_i = \sum_{t=1}^{N_r} \gamma^{t-1} r_{i,t}$. An action-value function is further defined in RL as an estimation of the total discounted reward:

$$Q_{\pi}(s_t, a_t) = E_{\pi} \left[\sum_{k=0}^{N_r} \gamma^k r_{t+k+1} \mid s_t, a_t \right] \quad (6.10)$$

In Eq. (6.10), the action-value function $Q_{\pi}(s_t, a_t)$ is equal to the expected return starting from state s_t , taking action a_t , and thereafter following policy π . The goal of RL is to find the optimal policy π^* that maximizes the action-value function:

$$Q^*(s_t, a_t) = \max_{\pi} Q_{\pi}(s_t, a_t) \quad (6.11)$$

One typical way for solving Eq. (6.11) is to update the action value based on the temporal difference (TD) error:

$$Q_{\pi}^{(k+1)}(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q_{\pi}^{(k)}(s_{t+1}, a_{t+1}) \quad (6.12)$$

In conventional RL methods such as Q-learning, a look-up table is established to store the action values of all the possible state-action pairs and it is updated iteratively according to (6.12) until convergence. However, the method encounters the curse of dimensionality when the state or action space becomes continuous. The deep RL method is developed to overcome the drawbacks of the tabular-based RL method. In deep RL, a neural network is designed to estimate the action-value function and it can form a continuous mapping between the state-action pair and the action value. In this way, more complex control or optimization problem with high dimensionality can be solved

through tweaking the neural network model.

6.3.2 Deep deterministic policy gradient method for continuous control

In this subsection, we will briefly introduce a deep RL method, DDPG, for solving continuous control problems.

In DDPG, there are two types of neural network: the critic network, and the actor network. The function of the critic network is to estimate the action value, as has been mentioned above. The input to the critic network is the current state and the action taken, and the output is the associated action value. The MSE is used as the loss function for updating the parameters of the critic network, as shown below:

$$Q^{target(j)}(t) = r^{(j)}(t) + \gamma \max_{a^{(j)}(t+1)} Q(s^{(j)}(t+1), a^{(j)}(t+1); \theta^{Q'}) \quad (6.13)$$

$$L(\theta^Q) = 1/N_s \sum_{j=1}^{N_s} (Q^{target(j)}(t) - Q(s^{(j)}(t), a^{(j)}(t); \theta^Q))^2 \quad (6.14)$$

Two critic networks are involved for calculating the MSE in (6.14), the target critic network, whose weights are noted as $\theta^{Q'}$; and the behavior critic network, whose weights are noted as θ^Q . In Eq. (6.13), the target action value at time step t is the sum of the current reward $r(t)$ and the discounted value of the maximum action value at the next time step $t+1$, generated by the target critic network. The superscript j is the index of state-action pair samples. Then, the target action value is sent to (6.14) for calculating the loss. The output from the target critic network is served as the “labelled” data for the behavior network to learn. During the training, the target critic network is updated at a slower speed than the behavior critic network, which helps stabilize the learning process.

The actor network is designed to utilize the estimated action value to obtain the optimal policy, i.e. $\pi(s(t)) = \operatorname{argmax}_{a(t)} Q(s(t), a(t))$ for all time step t . The input to the actor network is the current state $s(t)$, and the output is the action $a(t)$ that results in the maximum $Q(s(t), a(t))$. To achieve this goal, the loss function for the actor network is designed as follows:

$$\max J(\theta^\mu) = \frac{1}{N_s} \sum_{j=1}^{N_s} Q(s^{(j)}(t), a^j(t); \theta^Q) \big|_{a^j(t)=\mu(s^{(j)}(t); \theta^\mu)} \quad (6.15)$$

In Eq. (6.15), $\mu(s(t); \theta^\mu)$ is the current policy generated by the actor network, where θ^μ is the network weights. θ^μ is updated in the direction of maximizing the Q value using a gradient:

$$\nabla J(\theta^\mu) = \frac{1}{N_s} \sum_{j=1}^{N_s} \nabla Q_\mu(s^{(j)}(t), a^j(t); \theta^Q) \nabla_{\theta^\mu} \mu(s^{(j)}(t); \theta^\mu) \quad (6.16)$$

$$\theta^\mu = \theta^\mu - \eta_\mu \nabla J(\theta^\mu) \quad (6.17)$$

In Eq.(6.16), the chain rule is applied to calculate the gradient of the action value to the weights of the actor network. In Eq.(6.17), η_μ is the learning rate.

The above introduction covers the basic idea behind the DDPG algorithm. Note that in the above actor and critic network, only the action and the Q value at the current state is generated, and there's no need to store all the possible state-action pairs and their action values. The relationship between the state-action pair and the action value is encoded in the weights of the neural network. Therefore, DDPG can be applied for optimizing continuous control strategies without suffering from the dimensionality explosion.

6.3.3 MADDPG for solving Markov game in day-ahead electricity market bidding

Section 6.3.2 introduces the DDPG method for continuous control, which can be applied for optimizing the single agent decision-making process. However, in the case of day-ahead electricity market bidding, where multiple strategic GENCO bidders are involved, directly applying the above DDPG method for each GENCO cannot achieve the ideal results. This is because when multiple agents are optimizing their decisions simultaneously, the environment becomes dynamic, and the reward received at the same state with the same action can constantly change due to the changing policies of other agents. This issue invalidates the experience learnt by the target critic network and can result in incorrect target value settings and algorithm divergence.

Driven by the above concerns, there have been some recent research works in AI that include

multi-agent (MA) learning as an extension of the DDPG method to form MADDPG [147]. The main idea of MADDPG is to implement a centralized training, where the input to the critic network includes not only the state and action of the current agent, but also the actions of other agents. This assumption is acceptable because the critic network is only required during the training process. Once the algorithm is well-trained, only the actor network is needed for testing in new environments, and the actions of other agents are no longer required.

In this paper, the general-purposed MADDPG algorithm is applied and customized for solving the Markov game in day-ahead market bidding. The proposed customized MADDPG algorithm flow is shown in Algorithm 1 below.

Algorithm 1: MADDPG algorithm for day-ahead market bidding with N GENCO bidders

- 1: Initialize the parameters of the critic network $Q(s, a; \theta^Q)$ and the actor network $\mu(o; \theta^\mu)$ for each GENCO bidder
 - 2: Initialize the target networks with θ^Q and θ^μ
 - 3: **for** episode = 1 to M **do**
 - 4: Initialize the market bidding from a random day
 - 5: **for** $t = 1$ to N_T **do**
 - 6: Observe the current state $s(t) = [P_{load}(t), \lambda_1^{bid}(t-1), \lambda_2^{bid}(t-1), \dots, \lambda_g^{bid}(t-1), \dots, \lambda_N^{bid}(t-1)]$
 - 7: For each GENCO bidder g , select the bidding price $\lambda_g^{bid}(t) = \mu_g(o_g(t); \theta^\mu)$, where $o_g(t) = [P_{load}(t), \lambda_g^{bid}(t-1)]$
 - 8: Run DCOPF (1)-(5) to complete the market clearing, obtain the cleared quantity $P_g^{cleared}(t)$, cleared price $\lambda_g^{cleared}(t)$, and the reward $r_g(t)$ for each GENCO, and observe the next state $[P_{load}(t+1), \lambda_g^{bid}(t)]$
 - 9: Store the transition $(s(t), \lambda_g^{bid}(t), \lambda_g^{bid-}(t), r_g(t), s(t+1))$ for each GENCO
 - 10: **for** GENCO $g = 1$ to N **do**
 - 11: Randomly sample a minibatch of S samples $(s^{(j)}(t), \lambda_g^{bid(j)}(t), \lambda_g^{bid-(j)}(t), r_g^{(j)}(t), s^{(j)}(t+1))$ from the stored transitions
 - 12: Set $Q_g^{target(j)}(t) = r_g^{(j)}(t) + \gamma Q_g(s^{(j)}(t+1), \lambda_g^{bid(j)}(t+1), \lambda_g^{bid-(j)}(t+1); \theta^Q)$, for $\lambda_g^{bid(j)}(t+1) = \mu_g(o_g^{(j)}(t+1); \theta^\mu)$
 - 13: Update the critic network by minimizing the MSE:
 - 14: $L_g(\theta^Q) = 1/N_s \sum_j (Q_g^{target(j)}(t) - \gamma Q_g(s^{(j)}(t), \lambda_g^{bid(j)}(t), \lambda_g^{bid-(j)}(t); \theta^Q))^2$
 - 15: $\theta^Q = \theta^Q - \eta_Q \nabla_{\theta^Q} L_g(\theta^Q)$
 - 16: Update the actor network by maximizing the expected Q value:
 - 17: $\nabla J_g(\theta^\mu) = 1/N_s \sum_j \nabla_{\mu_g} Q_g(s^{(j)}(t), \lambda_g^{bid(j)}(t), \lambda_g^{bid-(j)}(t); \theta^Q) \nabla_{\theta^\mu} \mu_g(o_g^{(j)}(t); \theta^\mu)$
 - 18: $\theta^\mu = \theta^\mu - \eta_\mu \nabla J_g(\theta^\mu)$
 - 19: **end for**
 - 20: Update the target network parameters for each GENCO:
 - 21: $\theta^{Q'} = (1 - \tau)\theta^Q + \tau \theta^Q$
 - 22: $\theta^{\mu'} = (1 - \tau)\theta^\mu + \tau \theta^\mu$
 - 23: **end for**
 - 24: **end for**
-

In algorithm 1, the state is defined as the hourly load and the bidding prices of all the agents in the previous hour; the private observation of each agent is defined as the hourly load and its bidding price at the previous hour, as shown by line 6-7. The reward $r_g(t)$ is the hourly power selling profit:

$$r_g(t) = \sum_{b=1}^{N_b} P_{g,b}^{cleared}(t) \cdot \lambda_g^{cleared}(t) - C_g \left(\sum_{b=1}^{N_b} P_{g,b}^{cleared}(t) \right) \quad (6.18)$$

The state is sent to the critic network for calculating the target action value, as shown by line 12. $\lambda_g^{bid-(j)}(t+1)$ represents the bidding prices of all GENCOs except for the g^{th} GENCO. Note that the bidding price for the next time step $t+1$ is generated by $\mu_g(\theta')$ instead of $\mu_g(\theta)$ in line 12. Like the target critic network, $\mu_g(\theta')$ represents the target actor network, which also aims to stabilize the training process.

After the weights of the behavior critic network and the behavior actor network are updated, as shown by line 15 and 18, the weights of the target critic network and the target actor network are updated accordingly at a slower speed, as shown by line 21-22, where τ has a value close to 1. The reason for this slow update is also to increase the stability of the learning.

To help readers achieve an easy and clear understanding of the MADDPG algorithm for a multi-agent day-ahead market bidding problem, an illustration of the algorithm is shown in Figure 6.1:

6.3.4 Baseline cases for evaluating the learning performance of MADDPG

Two baseline cases are designed as a comparison with MADDPG algorithm. In the first baseline case, all the generators will bid truthfully by submitting their true marginal cost. In the second baseline case, a value-based deep RL method, the DQN, is applied for optimizing the GENCOs' bidding strategies. In DQN, discretization of the continuous action domain is required to estimate the Q value of each possible action, which can limit the search space and may not lead to the optimal action. The following section will present the simulation results from both MADDPG and the two baselines for comparison and analysis.

6.4 Simulation Analysis

6.4.1 Test system description

The IEEE 30-bus system with 9 generators is applied as the transmission-level electricity market. The topology of the system is shown in Figure 6.2:

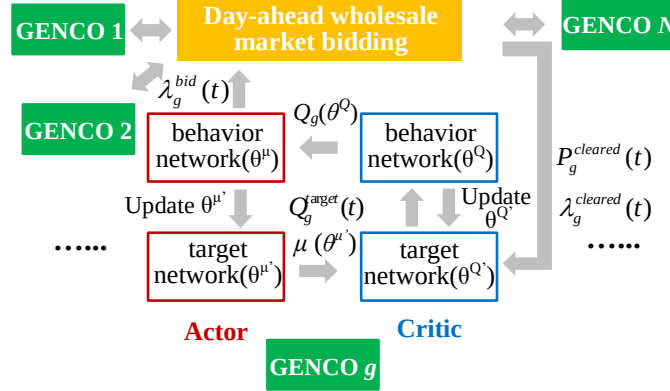


Figure 6.1 MADDPG algorithm for solving Markov game in day-ahead electricity market

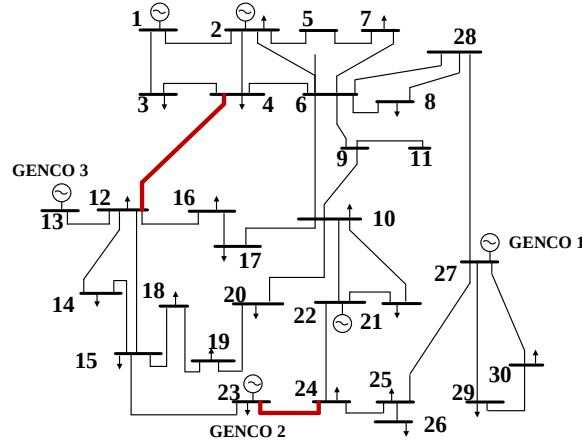


Figure 6.2. Topology of the IEEE 30-bus system

The generators at bus 27, bus 23, and bus 13 are considered as strategic bidders that will conduct economic withholding to maximize their profits. All other generators will submit their true marginal cost. In addition, the transmission lines 4-12 and 23-24 have a capacity limit of 10 MW. This capacity limit will give the nearby GENCOs, GENCO 2 and GENCO 3, the market power to manipulate the clearing price, which will later be shown in the simulation results.

The generation cost function of GENCOs is assumed to be a piecewise linear function, which include three segments. The parameters of the cost function are shown in Table 6.1:

Table 6.1 GENCO generation cost function

Segment	Marginal price (\$/MWh)	Generation range (MW)
1	20	[0-12]
2	40	[12-36]
3	50	[36-60]

It is assumed that at each hour, only one bidding block is submitted by each GENCO. The bidding quantity is 60 MW, which is their capacity. For GENCO 1-3, their bidding price is $\lambda_g^{bid}(t) = \varepsilon_g(t) \lambda_g^{cost}(t)$; for other GENCOs, their bidding price is $\lambda_g^{cost}(t)$. In this case, $\lambda_g^{cost}(t)$ is 50\$/MWh.

Following the assumptions presented at section 6.2.3, the value of the bidding price thresholds, th_1 and th_2 , are set to 1.5 and 1.1, respectively, which means $\lambda_g^{bid}(t)$ should comply with the following condition:

$$\frac{\max \lambda_g^{bid}(t)}{\min \lambda_g^{bid}(t)} \leq 1.5, 0.9 \leq \frac{\lambda_g^{bid}(t)}{\lambda_g^{bid}(t-1)} \leq 1.1, \forall t, \text{ for } g = 1, 2, 3 \quad (6.19)$$

6.4.2 Design of neural network and simulation platform

The detailed structures of actor network and critic network in the proposed MADDPG, as well as the structure of DQN are shown in Table 6.2:

Table 6.2. Design of DNNs in deep RL

Neural Network	Actor	Critic	DQN
Input	$[P_{load}(t), \lambda_g^{bid}(t-1)]$	$[P_{load}(t), \lambda_g^{bid}(t-1), \lambda_{vg}^{bid-}(t-1), \lambda_g^{bid}(t), \lambda_{vg}^{bid-}(t)]$	$[P_{load}(t), \lambda_g^{bid}(t-1)]$
No. of hidden layers	2	2	2
No. of neurons	[2,64],[64,64]	[7,64],[64,64]	[2,64]
Output	$\delta \in [0,1]$	$Q_g(s(t), \lambda_g^{bid}(t), \lambda_{vg}^{bid-}(t))$	$Q_g(s(t), \lambda_g^{bid}(t))$
Activation function	ReLU (hidden layer); sigmoid (output layer)	ReLU (hidden layer)	ReLU (hidden layer)
Learning rate(η)	0.001	0.002	0.001
τ	0.99	0.99	0.99
Optimizer	Adam	Adam	Adam

The output from the actor network is a value δ between 0 and 1. The bidding price $\lambda_g^{bid}(t)$ is calculated as follows:

$$\lambda_g^{bid}(t) = (0.9 + (1.1 - 0.9) \times \delta) \times \lambda_g^{bid}(t-1) \quad (6.20)$$

The value of $\lambda_g^{bid}(t)$ will be further adjusted to be within the range of $1 \sim 1.5 \lambda_g^{cost}(t)$:

$$\lambda_g^{bid}(t) = \begin{cases} 1.5 \lambda_g^{cost}(t), & \text{if } \lambda_g^{bid}(t) > 1.5 \lambda_g^{cost}(t) \\ \lambda_g^{cost}(t), & \text{if } \lambda_g^{bid}(t) < \lambda_g^{cost}(t) \end{cases} \quad (6.21)$$

$$\delta = (\frac{\lambda_g^{bid}(t)}{\lambda_g^{bid}(t-1)} - 0.9) / 0.2$$

For DQN, since the action $\lambda_g^{bid}(t)$ is within the continuous interval $1 \sim 1.5 \lambda_g^{cost}(t)$, a step size of 0.1 is applied to discretize the action space: $[1, 1.1, 1.2, 1.3, 1.4, 1.5] \times \lambda_g^{cost}(t)$. Hence, the output from DQN is a vector of size 1×6 , which includes the Q value for each potential $\lambda_g^{bid}(t)$.

The neural network model is built and trained by the open-source deep learning platform TensorFlow. The day-ahead market clearing process is completed by the smart market module in MATPOWER toolbox [57]. The hardware environment is a laptop with Intel®Core™ i7-7600U 2.8 GHz CPU, and 16.00 GB RAM.

6.4.3 Nash Equilibrium strategy from MADDPG: uncongested case

In this subsection, we first study the bidding strategies of the three GENCOs under marginal pricing mechanism in an uncongested case, where the capacity limit on line 4-12 and 23-24 are removed. The goal is to show that MADDPG algorithm is able to achieve the NE strategies that satisfy the two conditions as introduced in section 6.2.1, when no GENCO bidder has access to market power.

The load profile in June, 2019 from PJM wholesale market [133] is used to train the deep RL method. The load data in the 31 days in July, 2019 from PJM market is used to test the deep RL method after training. The load profile of the training days and the test days is shown in Figure 6.3, which shows the differences between the load levels in the two months. However, since the deep

RL is a generalized model, it can adapt to the changes in the environment and produces optimal strategies, which will be shown in the following test results.

Figure 6.4 presents the change of average reward over 500 training episode as a sign of algorithm convergence. As shown in the figure, the average rewards for the three GENCOs gradually stabilize as the training proceeds, which indicates the convergence of the MADDPG algorithm.

The test results of MADDPG with July data are shown in Figure 6.5 and are also compared with the truthful bidding baseline case in Table 6.3, where the 3 GENCOs always bids truthfully. In Figure 6.5, the y axis is the bidding parameter $\varepsilon_g(t)$ in Eq. (6.6). The red curve is the system load level. One thing should be pointed out is that since the state of GENCO bidder requires the bidding price at the previous hour, we assume that the bidding price at hour zero is always the true marginal cost. It can be observed that in the uncongested case, all GENCOs bid at their true marginal cost, regardless of the system load level. This is because when the capacity limit is removed, GENCO 2-3 cannot manipulate the market clearing price. Since all other GENCOs are bidding at their true marginal cost, the optimal bidding strategy for GENCO 1-3 is also truthful bidding. According to the NE conditions in section 6.2.1, v_g and b_g are $\lambda_g^{cost}(t)$ and $\lambda_g^{bid}(t)$, respectively. It can be seen that the truthful bidding, where $\lambda_g^{bid}(t) = \lambda_g^{cost}(t)$, satisfies the equality constraint in condition (i) and (ii). In Table 6.3, the total profit from MADDPG is the same as the baseline, since they both bid truthfully. Therefore, it can be safely concluded that the well-trained MADDPG algorithm can find the optimal bidding strategy of the three GENCO bidders in a constraint-free market environment.

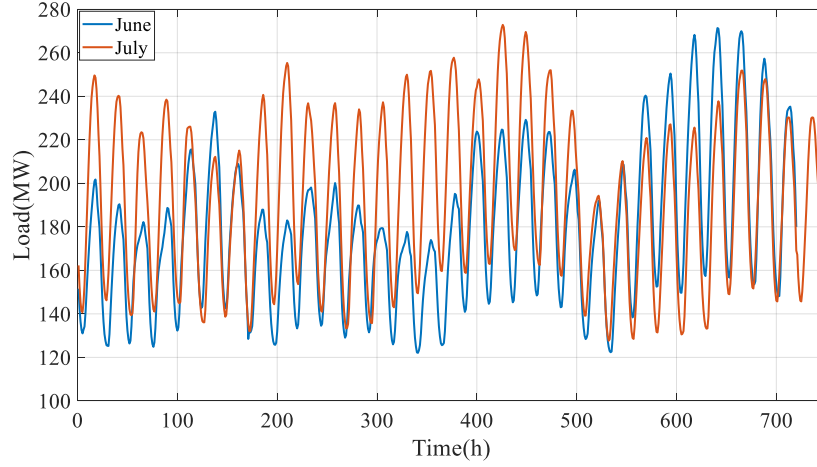


Figure 6.3. Load profile in June and July of year 2019

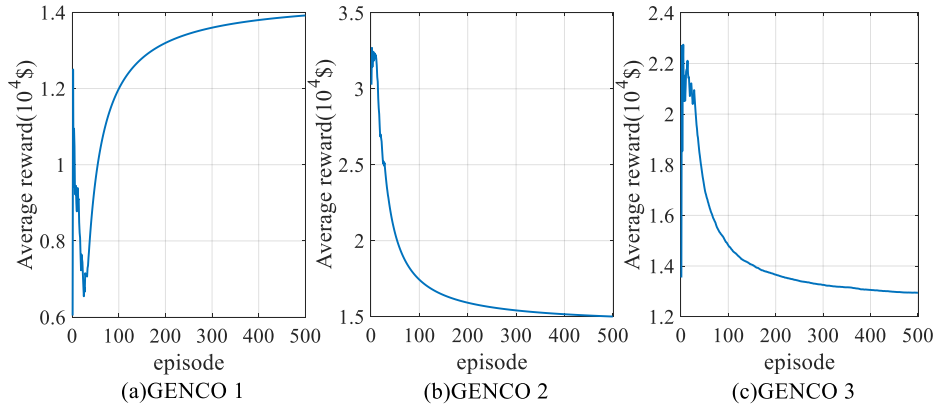


Figure 6.4. Convergence of MADDPG in the uncongested case

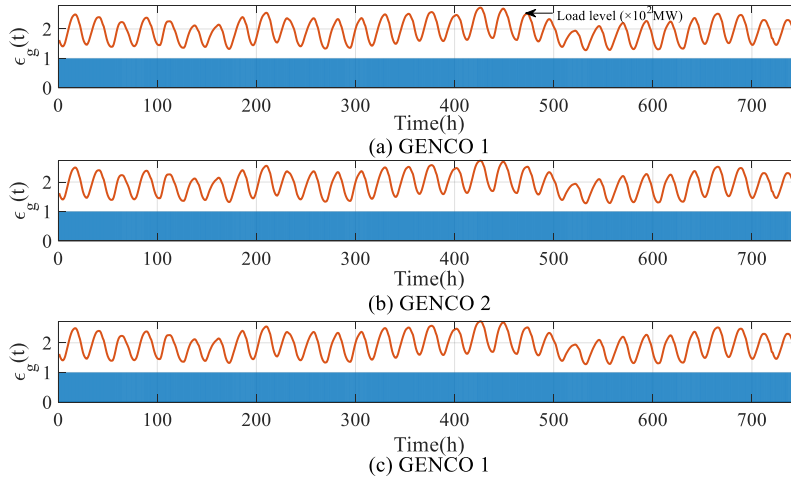


Figure 6.5. Bidding strategy of three GENCOs with MADDPG: uncongested case

Table 6.3. Comparison of MADDPG with baseline: uncongested case

GENCO	Total profit (10^4 \$)	
	MADDPG	Truthful bidding
1	44.64	44.64
2	44.64	44.64
3	40.26	40.26

6.4.4 Solving Markov game with MADDPG: congested case

In this subsection, the MADDPG algorithm is applied to solve the Markov game in day-ahead electricity market bidding with congestions. The same training data is used and the training results are shown in Figure 6.6: The algorithm is trained for 200 episodes. As shown in the figure, the average reward converges for all three GENCO bidders. The well-trained RL agents are then tested with the July data, and are also compared with the truthful bidding case, as shown in Figure 6.7- Figure 6.8, and Table 6.4.

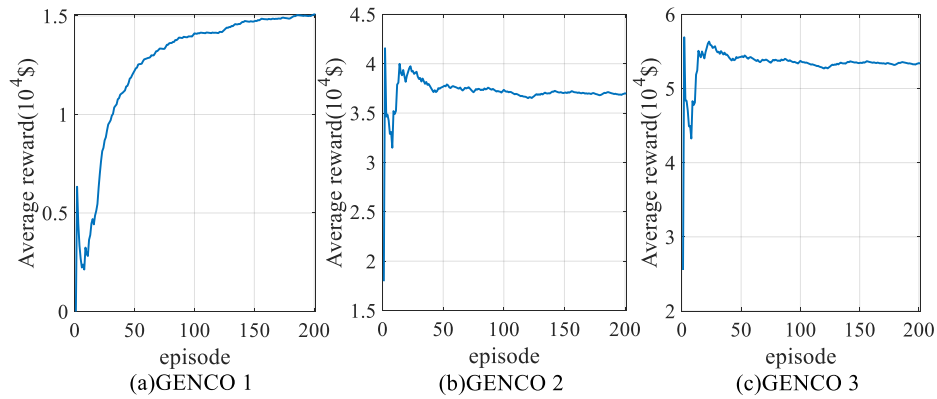


Figure 6.6. Convergence of MADDPG in the congested case

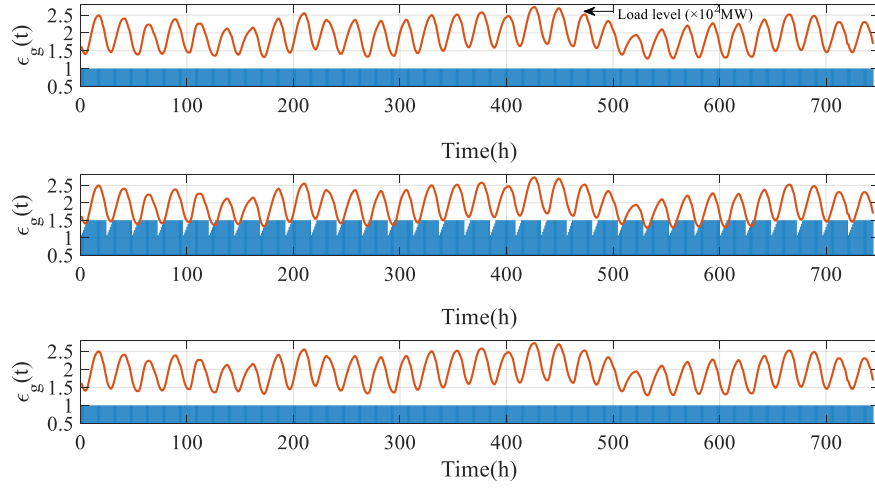


Figure 6.7. Bidding strategy of three GENCOs with MADDPG: congested case

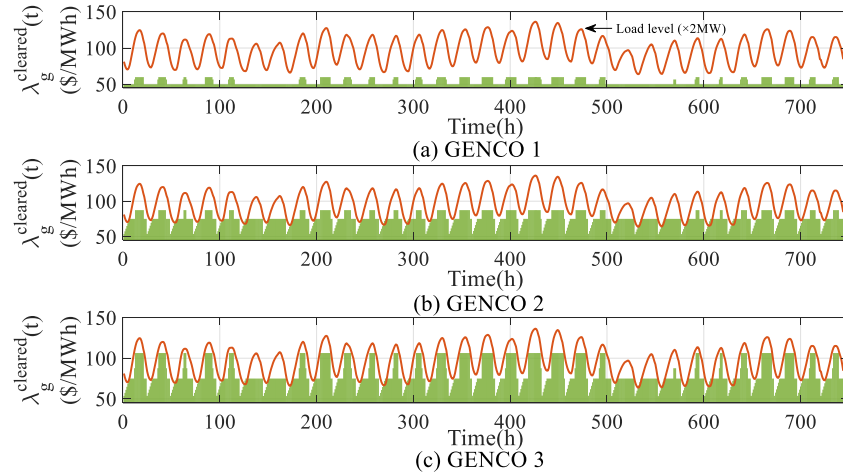


Figure 6.8. Hourly market clearing price under MADDPG bidding strategy

Table 6.4 Comparison of MADDPG with baselines: congested case

GENCO	Total profit (10^4 \$)		
	MADDPG	Truthful bidding	DQN
1	57.24	50.01	63.30
2	131.36	88.96	119.56
3	184.37	114.34	142.01

Figure 6.7 shows that GENCO 2 always bids at the highest price, which is $1.5 \lambda_g^{cost}(t)$, at the peak hours when congestions are most likely to happen. Since GENCO 1 does not have market power, it always bids at the true marginal cost, where $\varepsilon_g(t)$ is 1. GENCO 3 also bids at the true marginal cost. However, because the marginal price principle is applied for clearing the market, GENCOs 1 and 3 still benefit from the high bidding price offered by GENCO 2. As shown in Figure 6.8, the cleared prices for all three GENCOs are higher than their marginal generation cost 50\$/MWh. In Table 6.4, all three GENCOs obtain higher total profits than in the truthful bidding case. This phenomenon is called “free riding” in game theory, where GENCOs 1 and 3 can bid at a lower price to get more of their quantity cleared at the high marginal price.

The above bidding strategies form one NE and the reason is provided as follows: according to the definition of NE, no player can benefit by changing its strategy while the strategies of other players remain unchanged. First, since GENCO 1 has no market power, increasing its bidding price will only reduce its cleared quantity and profit. GENCO 1 will not bid higher than the true marginal cost; secondly, if GENCO 2 decreases the bidding price to the marginal cost, then all three GENCOs will bid truthfully like the baseline case, and all of them will receive a lower profit; lastly, if GENCO 3 also adopts a similar strategy like GENCO 2, which is to bid high at peak hours, then the amount of its cleared power will be greatly reduced, which results in a lower profit (this has been tested through simulation). Therefore, no GENCO will be willing to change its bidding strategy alone while the other two remain unchanged, which indicates a NE status.

The last column in Table 6.4 lists the total profit obtained by DQN-based bidding strategies. The average reward curve of DQN algorithm is shown in Figure 6.9. In DQN, an individual Q network is designed for each GENCO bidder, and there is no centralized training mechanism like the critic network in MADDPG. Therefore, it takes a longer time for the algorithm to converge. Figure 6.10 shows the bidding strategies derived from DQN for the test days. As shown, because of the limited search space and a lack of centralized training, the DQN algorithm generated a less profitable bidding strategy than the MADDPG algorithm. Note that although the profit of GENCO 1 is higher

with DQN, this is not a NE since GENCO 2 and GENCO 3 have the motivation to change their bidding strategies to obtain higher profits.

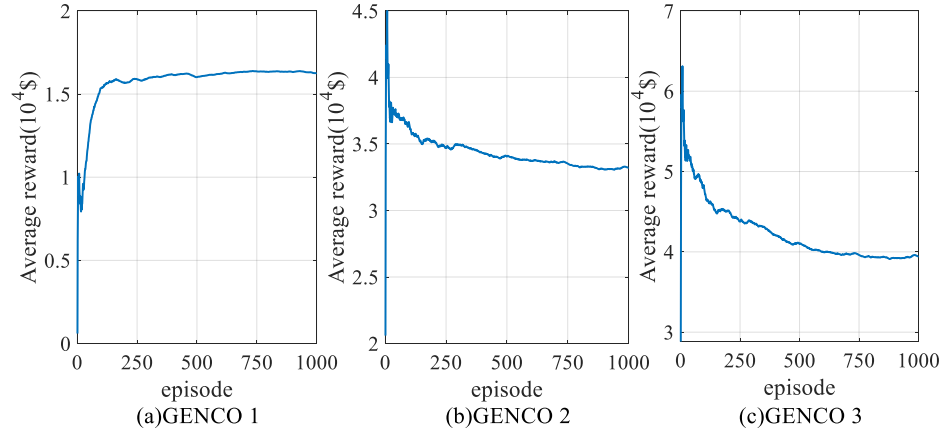


Figure 6.9. Convergence of DQN in the congested case

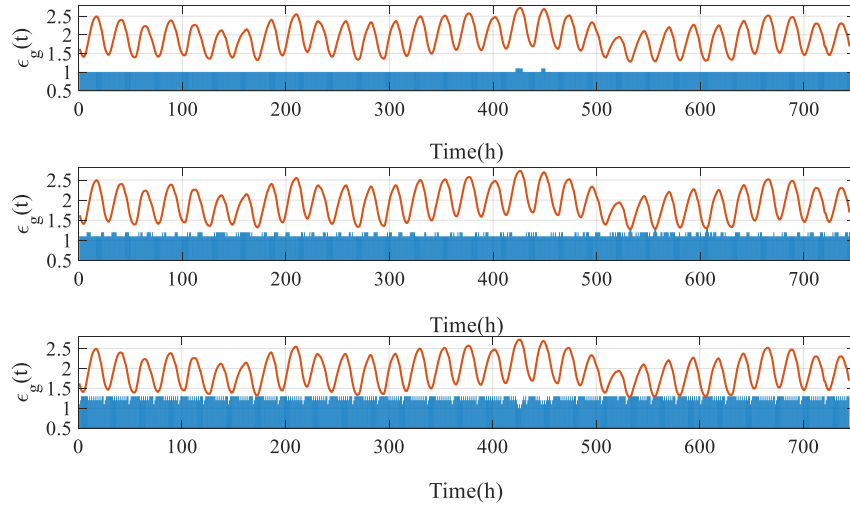


Figure 6.10. Bidding strategy of three GENCOs with DQN: congested case

6.5 Conclusions

This chapter presents an MADDPG algorithm for solving the Markov game in the day-ahead electricity market. The MADDPG algorithm can learn a profitable bidding strategy for multiple GENCO bidders through centralized training and decentralized execution. Simulation results verify the learning efficiency and computational efficiency of the MADDPG algorithm, which indicates that the algorithm can be a promising tool for solving power system problems that have multiple decision makers and high unobservability.

For future works, one potential direction is to study the sequential market bidding problem, where the bidding will take place in several market stages, including the day-ahead, intra-day, and real-time market. In such cases, MADDPG can still be applied to solve the associated Markov game through continuous interaction with the environment.

6.6 Acknowledgement

This work was supported in part by NSF Award under Grant ECCS-1809458, and in part by CURENT, a NSF/DOE Engineering Research Center through NSF Award under Grant EEC-1041877.

Chapter 7 Conclusions and Future Works

7.1 Summary of Current Progress

Deep learning technology has become a heated topic in both academia and industry. The rapid development of hardware platform provides a solid support for the realization of deep learning models in various application fields. It is foreseeable that with the ongoing acceleration of hardware processors, the performance of deep learning methods will be continuously improved and get adapted to more complex and intractable real-world situations and produce fruitful results.

In this dissertation, the implications and advancements of deep learning technology are first reviewed. Then, the current applications of deep learning technologies in the field of power systems are categorized and summarized. Based on the current research works, in this dissertation, several types of power system problems, that are either suffering from high computational burden, or mathematically difficult to model, are introduced, and the potential of applying deep learning methods for solving these problems is discussed.

In Chapter 2, a data-driven N-1 contingency screening method is applied for identifying the system security status under N-1 contingency with multiple uncertain operation scenarios. The deep CNN is utilized as a multi-task learning tool, which generates both system state variables and system security classification. Because of its high generalization ability, once the deep CNN is well trained, it can be directly applied to new operation scenarios and quickly estimates the system security status, without the iterative power flow calculation. The applied method can serve as an ancillary tool for power system security assessment in real-time operation.

Following the above work, in Chapter 3, a fast cascading outage screening method is applied for identifying the severest cascading outage path in the power system, which considers the sequential contingencies taking place. The method is a combination of deep CNN and DFS method. The deep CNN can estimate the value of the system security index via the training process. Then, based on the estimated value, the DFS method quickly scans all the potential cascading outage paths within

the system and identifies the paths with highest severity. The simulation results based on standard IEEE test cases verify the computational efficiency and the accuracy of the above data-driven method, which can provide reference for power system operators to take corrective measures to prevent system collapse in advance during real-time operation.

In Chapter 4, the DNN is applied for optimizing MMG energy management at the distribution system level. Due to the protection of user privacy, the microgrid is modeled as a black box, and the DNN is utilized to simulate the behavior of multiple microgrids under real-time prices. Then, with the well-trained DNN as a representative of MMG, Monte Carlo RL method is applied for the DSO to decide the optimal real-time price sequences so that the system PAR can be minimized, while maintaining a high profit of selling power for DSO. Case studies demonstrate that the applied data-driven method is highly computationally efficient compared with the conventional model-based method for solving microgrid economic operation, which proves its feasibility in real-world applications.

In Chapter 5, a data-driven, continuous control strategy for multi-zone residential HVAC system is proposed based on deep RL method. The DDPG algorithm is applied to explore the optimal selection of setpoints of the HVAC system to maintain the indoor temperature within the user's comfort level while minimizing the energy consumption cost. Within such a data-driven framework, the algorithm requires no detailed formulation of the complex thermal dynamic model within the HVAC system, but gradually learns the optimal control policy through continuous interaction with the environment. Comparisons with benchmark baseline cases verify that the applied deep RL based control algorithm is able to find a more economic HVAC control strategy while maintaining user's comfort. The well-trained deep RL agent has also gained considerable generalization and can be adapted to unseen environments with different physical conditions.

In Chapter 6, a multi-agent deep RL method is applied to solve the Markov game at the day-ahead electricity bidding market. The objective is to obtain an optimal bidding strategy for each intelligent GENCO bidder that will result in the largest individual profit, hence no player will

change its bidding strategy when other players remain unchanged, which leads to a NE. The MADDPG method is implemented for obtaining the NE status. The advantage of MADDPG over the DDPG is that the latter leverages a centralized-training, decentralized testing framework that considers the interactions among different decision-makers within a non-stationary environment. The simulation results on the IEEE 30-bus system in both an uncongested case and a congested case demonstrate that the applied MADDPG method is able to find a more economic bidding strategy for each GENCO bidder compared with a truthful bidding baseline case and also the widely applied DQN algorithm. It can be safely concluded that the MADDPG algorithm has substantial potential for solving complex decision-making problems with multiple decision-makers or hidden information.

7.2 Future Works

One major bottleneck of the current developed deep learning algorithms is the adaptability. Adaptability means that a well-trained deep learning algorithm can get quickly adapted to new tasks based on its past learning experience with the least retraining efforts, and generates results with guaranteed accuracy. This type of effective learning is a key capability of human beings, yet machine learning methods are still a long way from realization of such capability. It is crucial for deep learning algorithms to develop the adaptability in order for their continuous flourishing because under certain real-world situations there is no access to a large amount of training data for the algorithm to learn, which can lead to the failure of heavily data-dependent methods like supervised learning. In addition, training a machine learning algorithm can be tremendously time-consuming and resource-consuming, yet new tasks keep streaming in with different optimization objectives. As a consequence, it becomes unpractical to train a brand-new model for each individual task starting from scratch.

There have been extensive research efforts in literature dedicating to enhancing the adaptability of machine learning algorithms within the context of multi-task learning. Transfer learning and

Meta-learning are two of the latest research hot spots. Transfer learning focuses on how to utilize the knowledge from a source task to accelerate the learning process when solving a target task, which is different but related to the source task. Meta learning applies a more simple philosophy, which is known as learning to learn. In traditional machine learnings, the algorithm can formulate a mapping between the input data and the desired output after thousands of rounds of iterations such as back propagation. Then a new task comes and the above process has to be repeated to achieve a desired model. What if instead of simply learning the extracted features from the given data, we let the machine learning model learn how this mapping is achieved, for example, how the neural networks are initialized (normal distribution, Xavier initialization, etc.), and how the parameters of the neural network are updated (back-propagation). These key issues are like a general tool for solving any data-driven related task. If the machine learning algorithm can understand these core ideas, it will become possible to quickly adapt to unseen tasks by applying the same idea with reduced retraining efforts. There is an old Chinese saying that goes “Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime.” Meta learning adopts a similar principle: teaching the machine the generalized learning skills instead of simply offering the specific answers.

It is foreseeable that more complicated real-world tasks with high dimension or model inaccessibility will grow explosively in the near future, and the functionality of the heavily data-dependent machine learning methods will become more limited under such scenarios. It is highly imperative to make the best of deep learning techniques like transfer learning and meta-learning to design algorithms with enhanced robustness and extended adaptability, to fully prepare for unpredicted learning tasks within a perplexing environment. The exploration of the potential applications of transfer learning and meta-learning in the field of power system control and operation will be the key future research directions. Some potentially interesting research topics include:

- 1) Development of a more generalized contingency screening tool with deep CNN and transfer

learning that can be adapted to different types of contingencies and power systems at different scales, with least amount of retraining efforts;

2) Applying deep RL for power system emergency control under cascading outages to restore system security; further combining with transfer learning to learn a more robust control strategy that can be adapted to various fault scenarios;

3) Design of a multi-task learning framework based on deep RL and meta-learning that can optimally control a multi-zone residential HVAC system in both heating scenarios and cooling scenarios with high learning efficiency;

4) Utilization of deep learning and transfer learning to estimate power system transient stability under different system topologies and operation scenarios to accelerate the simulation process.

References

- [1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [2] I. Goodfellow, Y. Bengio, and A. Courville, “Deep Learning,” Cambridge: MIT press, 2016, pp. 9-12.
- [3] F. Chollet, “Deep Learning with Python,” Manning Publications Co., 2017, pp. 6.
- [4] M. D. Zeiler, R. Fergus, “Visualizing and Understanding Convolutional Networks,” arXiv: 1311. 2901, 2013.
- [5] H. Shi, M. Xu, and R. Li. “Deep learning for household load forecasting—A novel pooling deep RNN,” *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271-5280, Sep. 2018.
- [6] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, “Short-Term Residential Load Forecasting based on Resident Behaviour Learning,” *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 1087-1088, Jan. 2018.
- [7] M. Sun, T. Zhang, Y. Wang, G. Strbac, and C. Kang, “Using Bayesian Deep Learning to Capture Uncertainty for Residential Net Load Forecasting,” *IEEE Trans. Power Syst.*, 2019, early access.
- [8] M. Khodayar, and J. Wang, “Spatio-temporal graph deep neural network for short-term wind speed forecasting,” *IEEE Trans. Sustainable Energy*, vol. 10, no. 2, pp. 670-681, Apr. 2019.
- [9] Q. Zhu, J. Chen, D. Shi, L. Zhu, X. Bai, X. Duan, and Y. Liu, “Learning Temporal and Spatial Correlations Jointly: A Unified Framework for Wind Speed Prediction,”. *IEEE Trans. Sustainable Energy*, 2019, early access.
- [10] J. F. Toubreau, J. Bottieau, F. Vallée, and Z. De Grève, “Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets,” *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1203-1215, Mar. 2019.
- [11] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, J. He, “Short-term load forecasting with deep residual networks,” *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943-3952, Jul. 2019.
- [12] W. Zhang, H. Quan, and D. Srinivasan, “An improved quantile regression neural network for probabilistic load forecasting,” *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4425-4434, Jul. 2019.
- [13] W. Kong, Z. Dong, B. Wang, J. Zhao, and J. Huang, “A Practical Solution for Non-Intrusive Type II Load Monitoring based on Deep Learning and Post-Processing,” *IEEE Trans. Smart Grid*, 2019, early access.
- [14] V. Singhal, J. Maggu, and A. Majumdar, “Simultaneous detection of multiple appliances from smart-meter measurements via multi-label consistent deep dictionary learning and deep transform learning,” *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 2969-2978, May 2019.
- [15] M. Cui, M. Khodayar, C. Chen, X. Wang, Y. Zhang, and M. E. Khodayar, “Deep Learning Based Time-Varying Parameter Identification for System-Wide Load Modeling,” *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp 6102-6114, Nov. 2019.

- [16]Y. Wang, Q. Chen, D. Gan, J. Yang, D. S. Kirschen, C. Kang, “Deep learning-based socio-demographic information identification from smart meter data,” *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 2593-2602, May 2019.
- [17]Z. Zheng, Y. Yang, X. Niu, H. Dai, and Y. Zhou, “Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids,” *IEEE Trans. Industrial Informatics*, vol. 14, no. 4, pp. 1606-1615, Apr. 2018.
- [18]K. L. López, C. Gagné, and M. A. Gardner, “Demand-side management using deep learning for smart charging of electric vehicles,” *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 2683-2691, May 2019.
- [19]S. Wen, Y. Wang, Y. Tang, Y. Xu, P. Li, and T. Zhao, “Real-Time Identification of Power Fluctuations based on LSTM Recurrent Neural Network: A Case Study on Singapore Power System,” *IEEE Trans. Industrial Informatics*, vol. 15, no. 9, pp. 5266-5275, Sep. 2019.
- [20]X. Wang, H. He, and L. Li, “A Hierarchical Deep Domain Adaptation Approach for Fault Diagnosis of Power Plant Thermal System,” *IEEE Trans. Industrial Informatics*, vol. 15, no. 9, pp. 5139-5148, Sep. 2019.
- [21]W. Li, D. Deka, M. Chertkov, and M. Wang, “Real-time Faulted Line Localization and PMU Placement in Power Systems through Convolutional Neural Networks,” *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4640-4651, Nov. 2019.
- [22]R. Yadav, S. Raj, and A. K. Pradhan, “Real-time Event Classification in Power System with Renewables using Kernel Density Estimation and Deep Neural Network,” *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6849-6859, Nov. 2019.
- [23]A. Gupta, G. Gurralla, and P. S. Sastry, “A Deep Learning-Based Feature Extraction Framework for System Security Assessment,” *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 864-872, Mar. 2019.
- [24]M. Sun, I. Konstantelos, and G. Strbac, “A deep learning-based feature extraction framework for system security assessment,” *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5007-5020, Sep. 2019.
- [25]X. Kong, X. Xu, Z. Yan, S. Chen, H. Yang, and D. Han, “Deep learning hybrid method for islanding detection in distributed generation,” *Applied Energy*, vol. 210, pp. 776-785, Jan. 2018.
- [26]R. Yan, G. Geng, Q. Jiang, and Y. Li, “Fast Transient Stability Batch Assessment using Cascaded Convolutional Neural Networks,” *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 2802-2813, Jul. 2019.
- [27]Y. He, G. J. Mendis, and J. Wei, “Real-Time Detection of False Data Injection Attacks in Smart Grid: A Deep Learning-Based Intelligent Mechanism,” *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2505-2516, Sep. 2017.
- [28]K. R. Mestav, J. Luengo-Rozas, and L. Tong, “Bayesian State Estimation for Unobservable Distribution Systems via Deep Learning,” *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4910-4920, Nov. 2019.

- [29]M. Sun, Y. Wang, F. Teng, Y. Ye, G. Strbac, and C. Kang, "Clustering-Based Residential Baseline Estimation: A Probabilistic Perspective," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6014-6028, Nov. 2019.
- [30]H. Jahangir, H. Tayarani, S. Baghali, A. Ahmadian, A. Elkamel, M. A. Golkar, and M. Castilla, "A Novel Electricity Price Forecasting Approach Based on Dimension Reduction Strategy and Rough Artificial Neural Networks," *IEEE Trans. Industrial Informatics*, 2019, early access.
- [31]L. Wang, Z. Zhang, and J. Chen, "Short-term electricity price forecasting with stacked denoising autoencoders," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 2673-2681, Jul. 2017.
- [32]J. Yan, H. Zhang, Y. Liu, S. Han, L. Li, and Z. Lu, "Forecasting the high penetration of wind power on multiple scales using multi-to-multi mapping," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3276-3284, May. 2018.
- [33]X. Luo, J. Sun, L. Wang, et al. "Short-term wind speed forecasting via stacked extreme learning machine with generalized correntropy," *IEEE Trans. Industrial Informatics*, vol. 14, no. 11, pp. 4963-4971, Nov. 2018.
- [34]M. Khodayar, S. Mohammadi, M. E. Khodayar, J. Wang, and G. Liu, "Convolutional Graph Autoencoder: A Generative Deep Neural Network for Probabilistic Spatio-temporal Solar Irradiance Forecasting," *IEEE Trans. Sustainable Energy*, 2019, early access.
- [35]Y. Zhang, V. V. G. Krishnan, J. Pi, K. Kaur, A. Srivastava, A. Hahn, and S. Suresh, "Cyber physical security analytics for transactive energy systems," *IEEE Trans. Smart Grid*, 2019, early access.
- [36]R. Sun, Y. Liu, and L. Wang, "An Online Generator Start-Up Algorithm for Transmission System Self-Healing Based on MCTS and Sparse Autoencoder," *IEEE Trans. Power Syst.*, vol. 34, no. 3, pp. 2061-2070, May. 2019.
- [37]Y. Chen, Y. Wang, D. Kirschen, and B. Zhang, "Model-Free Renewable Scenario Generation Using Generative Adversarial Networks," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3265-3275, May 2018.
- [38]C. Ren, and Y. Xu, "A fully data-driven method based on generative adversarial networks for power system dynamic security assessment with missing data," *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 5044-5052. Nov. 2019.
- [39]B. J. Claessens, P. Vrancx, and F. Ruelens, "Convolutional Neural Networks for Automatic State-Time Feature Extraction in Reinforcement Learning Applied to Residential Load Control," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3259-3269, Jul. 2018.
- [40]E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, G. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE Trans. Smart Grid*, vol. 10, no. 4, Jul. 2019.
- [41]P. Zeng, H. Li, H. He, and S. Li, "Dynamic Energy Management of a Microgrid using Approximate Dynamic Programming and Deep Recurrent Neural Network Learning," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4435-4445, Jul. 2019.

- [42]V. H. Bui, A. Hussain, and H. M. Kim, "Double Deep Q-Learning-Based Distributed Operation of Battery Energy Storage System Considering Uncertainties," *IEEE Trans. Smart Grid*, 2019, early access.
- [43]Z. Wan, H. L. H. He, and D. Prokhorov, "Model-Free Real-Time EV Charging Scheduling Based on Deep Reinforcement Learning," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5246-5257, Sep. 2019.
- [44]H. Xu, H. Sun, D. Nikovski, S. Kitamura, K. Mori, and H. Hashimoto, "Deep Reinforcement Learning for Joint Bidding and Pricing of Load Serving Entity," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6366-6375, Nov. 2019.
- [45]Z. Yan, and Y. Xu, "Data-Driven Load Frequency Control for Stochastic Power Systems: A Deep Reinforcement Learning Method with Continuous Action Search," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1653-1656, Mar. 2019.
- [46]N. Dagmar, and A. J. Germond, "Power system static security assessment using the Kohonen neural network classifier," *IEEE Trans. Power Syst.*, vol. 7, pp. 865-872, May 1992.
- [47]F. Li and Y. Du, "From AlphaGo to Power System AI: What Engineers Can Learn from Solving the Most Complex Board Game," *IEEE Power & Energy Magazine*, vol. 16, no. 2, pp. 76-84, Mar. 2018.
- [48]N. Yorino, M. Abdillah, Y. Sasaki, and Y. Zoka, "Robust Power System Security Assessment Under Uncertainties Using Bi-Level Optimization," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 352-362, Jan. 2018.
- [49]Y. Yang, X. Guan, and Q. Zhai, "Fast Grid Security Assessment with $N - k$ Contingencies," *IEEE Trans. Power Syst.*, vol. 32, no. 3, pp. 2193-2203, May 2017.
- [50]D. A. Tejada-Arango, P. Sánchez-Martín, and A. Ramos, "Security constrained unit commitment using line outage distribution factors," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 329-337, Jan. 2018.
- [51]M. Majidi-Qadikolai, R. Baldick, "Stochastic Transmission Capacity Expansion Planning with Special Scenario Selection for Integrating N-1 Contingency Analysis," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4901-4912, Nov. 2016.
- [52]P. Liu, Z. Li, Y. Zhuo, X. Lin, S. Ding, M. S. Khalid, and O. S. Adio, "Design of Wind Turbine Dynamic Trip-Off Risk Alarming Mechanism for Large-Scale Wind Farms," *IEEE Trans. Sustainable Energy*, vol. 8, no. 4, pp. 1668-1678, Oct 2017.
- [53]R. Sunitha, R. S. Kumar, and A. T. Mathew, "Online static security assessment module using artificial neural networks," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4328-4335, Nov. 2013.
- [54]S. Kalyani, and K. Shanti Swarup, "Classification and Assessment of Power System Security Using Multiclass SVM," *IEEE Trans. Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 5, pp. 753-758, Sep. 2011.
- [55]K. Nara, K. Tanaka, and H. Kodama, etc. "On-line contingency selection algorithm for voltage security analysis," *IEEE Trans. Power Apparatus and Systems*, vol. 4, pp. 846-856, Apr. 1985.

- [56]R. Sunitha, R. Sreerama Kumar, and A. T. Mathew, “Development of a Composite Security Index for Static Security Evaluation,” in TENCON 2009-2009 IEEE Region 10 Conference.
- [57]R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, “MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education,” *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12-19, Feb. 2011
- [58]Glossary of Terms Used in NERC Reliability Standards. [Online]. Available: https://www.nerc.com/files/glossary_of_terms.pdf
- [59]WSCC Operations Committee, “Western Systems Coordinating Council Disturbance Report for the Power System Outages that Occurred on the Western Interconnection on July 2, 1996 and July 3, 1996,” Western Syst. Coordinating Council, Salt Lake City, UT, USA, Tech. Rep., 1996.
- [60]U.S.-Canada Power System Outage Task Force, “Final report on the August 14th blackout in the United States and Canada,” Apr. 2004.
- [61]Protection System Response to Power Swings, System Protection and Control Subcommittee, NERC, Atlanta, GA, USA, Aug. 2013. [Online]. Available: <http://www.nerc.com>
- [62]Transmission System Planning Performance Requirements, NERC Standard TPL-001-4, 2013.
- [63]I. Dobson, B.A. Carreras, V.E. Lynch, and D.E. Newman, “Complex systems analysis of series of blackouts: cascading failure, critical points, and self-organization,” *Chaos*, vol. 17, 026103, June 2007.
- [64]S. Mei, Y. Ni, Weng, G. Wang, and S. Wu, “A study of self-organized criticality of power system under cascading failures based on AC-OPA with voltage stability margin,” *IEEE Trans. Power Syst.*, vol. 23, no. 4, pp. 1719–1726, Nov. 2008.
- [65]D. S. Kirschen, D. Jawayeera, D. P. Nedic, and R. N. Allan, “A probabilistic indicator of system stress,” *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 1650–1657, Aug. 2004.
- [66]I. Dobson, B. A. Carreras, and D. E. Newman, “A loading dependent model of probabilistic cascading failure,” *Probability in the Engineering and Informational Sciences*, vol. 19, no. 1, pp. 15–32, Jan. 2005.
- [67]P. Henneaux, P.-E. Labeau, J.-C. Maun, and L. Haarla, “A Two-Level Probabilistic Risk Assessment of Cascading Outages,” *IEEE Trans. Power Syst.*, vol. 31, no. 3, pp. 2393-2403, 2016.
- [68]R. Yao, S. Huang, K. Sun, F. Liu, X. Zhang, and S. Mei, “A Multi-Timescale Quasi-Dynamic Model for Simulation of Cascading Outages,” *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 3189-3201, 2016.
- [69]J. Guo, F. Liu, J. Wang, J. Lin, and S. Mei, “Toward Efficient Cascading Outage Simulation and Probability Analysis in Power Systems,” *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 2370-2382, May 2018.
- [70]M. Rahnamay-Naeini, and M. M. Hayat, “Cascading Failures in Interdependent Infrastructures: An Interdependent Markov-Chain Approach,” *IEEE Trans. Smart Grid*, vol. 7, no. 4, pp. 1997-2006, Jul. 2016.
- [71]P. Hines, I. Dobson, and P. Rezaei, “Cascading Power Outages Propagate Locally in an Influence Graph that is not the Actual Grid Topology,” *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 958-967, Mar. 2017.

- [72]J. Qi, W. Ju, and K. Sun, "Estimating the Propagation of Interdependent Cascading Outages with Multi-Type Branching Processes," *IEEE Trans. Power Syst.*, vol. 32, no. 2, pp. 1212-1223, Mar. 2017.
- [73]J. Qi, J. Wang, and K. Sun, "Efficient Estimation of Component Interactions for Cascading Failure Analysis by EM Algorithm," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 3153-3161, May 2018.
- [74]R. Yao, S. Huang, K. Sun, F. Liu, X. Zhang, S. Mei, W. Wei, and L. Ding, "Risk Assessment of Multi-Timescale Cascading Outages Based on Markovian Tree Search," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 2887-2900, Jul. 2017.
- [75]R. Yao, and K. Sun, "Toward Simulation and Risk Assessment of Weather-Related Outages," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4391-4400, Jul. 2019.
- [76]J. Guo, F. Liu, J. Wang, M. Cao, and S. Mei, "Quantifying the Influence of Component Failure Probability on Cascading Blackout Risk," *IEEE Trans. Power Syst.*, vol. 33, no. 5, pp. 5671-5681, Sep. 2018.
- [77]X. Liu, and Z. Li, "Revealing the Impact of Multiple Solutions in DCOPF on the Risk Assessment of Line Cascading Failure in OPA Model," *IEEE Trans. Power Syst.*, vol. 31, no. 5, pp. 4159-4160, Sep. 2016.
- [78]Z. Ma, C. Shen, F. Liu, and S. Mei, "Fast Screening of Vulnerable Transmission Lines in Power Grids: A PageRank-Based Approach," *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 1982-1991, Mar. 2019.
- [79]X. Wei, J. Zhao, T. Huang, and E. Bompard, "A Novel Cascading Faults Graph Based Transmission Network Vulnerability Assessment Method," *IEEE Trans. Power Syst.*, vol. 33, no. 3, pp. 2995-3000, May 2018.
- [80]E. J. Anderson, and J. Linderoth, "High Throughput Computing for Massive Scenario Analysis and Optimization to Minimize Cascading Blackout Risk," *IEEE Trans. Smart Grid*, vol. 8, no. 3, pp. 1427-1435, May 2017.
- [81]A. A. Babalola, R. Belkacemi, and S. Zarrabian, "Real-Time Cascading Failures Prevention for Multiple Contingencies in Smart Grids Through a Multi-Agent System," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 373-385, Jan. 2018.
- [82]R. Yao, K. Sun, F. Liu, and S. Mei, "Management of Cascading Outage Risk Based on Risk Gradient and Markovian Tree Search," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 4050-4060, Jul. 2018.
- [83]M. R. Salimian, and M. R. Aghamohammadi, "A Three Stages Decision Tree-Based Intelligent Blackout Predictor for Power Systems Using Brittleness Indices," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5123-5131, Sep. 2018.
- [84]Y. Jia, Z. Xu, L. L. Lai, and K. P. Wong, "Risk-Based Power System Security Analysis Considering Cascading Outages," *IEEE Trans. Industrial Informatics*, vol. 12, no. 2, pp. 872-882, Apr. 2016.
- [85]Y. Du, F. Li, J. Li, and T. Zheng, "Achieving 100x Acceleration for N-1 Contingency Screening with Uncertain Scenarios using Deep Convolutional Neural Network," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 3303-3305, Jul. 2019.

- [86]X. Fang, Q. Hu, F. Li, B. Wang, and Y. Li, "Coupon-based demand response considering wind power uncertainty: A strategic bidding model for load serving entities," *IEEE Trans. Power Syst.*, vol. 31, no. 2, pp. 1025-1037, Mar. 2015.
- [87]Y. Du, and F. Li, "A Hierarchical Real-time Balancing Market Considering Multi-microgrids with Distributed Sustainable Resources," *IEEE Trans. Sustainable Energy*, vol. 11, no. 1, pp. 72-83, Jan. 2020.
- [88]J. P. Catalão, P. Siano, F. Li, M. A. Masoum, and J. Aghaei, "Guest Editorial Special Section on Industrial and Commercial Demand Response," *IEEE Trans. Industrial Informatics*, vol. 14, no. 11, pp. 5017-5019, Nov. 2018.
- [89]H. Shin, and R. Baldick, "Plug-in electric vehicle to home (V2H) operation under a grid outage," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 2032-2041, Jul. 2017.
- [90]A.-H. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 320-331, Nov. 2010.
- [91]R. Deng, Z. Yang, J. Chen, N. R. Asr, and M.-Y. Chow, "Residential Energy Consumption Scheduling: A Coupled-Constraint Game Approach," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1340-1350, May 2014.
- [92] H. M. Soliman, and A. Leon-Garcia, "Game-Theoretic Demand-Side Management With Storage Devices for the Future Smart Grid," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1475-1485, May 2014.
- [93] P. Samadi, H. Mohsenian-Rad, V. W. S. Wong, and R. Schober, "Real-Time Pricing for Demand Response Based on Stochastic Approximation," *IEEE Trans. Smart Grid*, vol. 5, no. 2, pp. 789-798, Mar. 2014.
- [94] C. Li, X. Yu, W. Yu, G. Chen, and J. Wang, "Efficient Computation for Sparse Load Shifting in Demand Side Management," *IEEE Trans. Smart Grid*, vol. 8, no. 1, pp. 250-261, Jan. 2017.
- [95] S. Bahrami, V. W. S. Wong, and J. Huang, "An Online Learning Algorithm for Demand Response in Smart Grid," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4712-4725, Sep. 2018.
- [96] L. Wang, Z. Zhang, and J. Chen. "Short-term electricity price forecasting with stacked denoising autoencoders," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 2673-2681, Jul. 2017.
- [97] D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra, "Residential demand response using reinforcement learning," in *2010 First IEEE International Conference on Smart Grid Communications*, 2010, pp. 409-414.
- [98] Z. Wen, D. O'Neill, and H. Maei, "Optimal Demand Response Using Device-Based Reinforcement Learning," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2312-2324, Sep. 2015.
- [99] A. Sheikhi, M. Rayati, and A. M. Ranjbar, "Dynamic load management for a residential customer; reinforcement learning approach," *Sustainable Cities and Society*, vol. 24, pp. 42-51, Jul. 2016.

- [100]M. Parvania, and M. Fotuhi-Firuzabad, "Demand response scheduling by stochastic SCUC," *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 89-98, Jun. 2010.
- [101]W. Liu, P. Zhuang, H. Liang, J. Peng, Z. Huang, "Distributed Economic Dispatch in Microgrids Based on Cooperative Reinforcement Learning," *IEEE Trans. Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2192-2203, Jun. 2018.
- [102]R. S. Sutton, and A. G. Barto, "Reinforcement Learning: An Introduction," Cambridge: MIT press, 2018.
- [103]H. Yuan, F. Li, Y. Wei, and J. Zhu, "Novel Linearized Power Flow and Linearized OPF Models for Active Distribution Networks with Application in Distribution LMP," *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 438-448, Jan. 2018.
- [104]D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint, arXiv:1412.6980, 2014.
- [105]X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.
- [106]S. Ioffe, C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint, arXiv:1502.03167, 2015.
- [107]L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and buildings*, vol. 40, no. 3, pp. 394-398, 2008.
- [108]A. Costa, M. M. Keane, J. I. Torrens, and E. Corry, "Building operation and energy performance: Monitoring, analysis and optimisation toolkit," *Applied Energy*, vol. 101, pp. 310-316, 2013.
- [109]X. Kou, F. Li, J. Dong, M. Starke, J. Munk, T. Kuruganti, and H. Zandi, "A Distributed Energy Management Approach for Residential Demand Response," in the 3rd International Conference on SMART GRID AND SMART CITIES (ICSGSC), Berkeley, CA, Jun. 25-28, 2019.
- [110]K. Ma, G. Hu, and C. J. Spanos, "Energy management considering load operations and forecast errors with application to HVAC systems," *IEEE Trans. Smart Grid*, vol. 9, pp. 605-614, Mar. 2018.
- [111]O. Erdinc, A. Taşçıkaraoğlu, N. G. Paterakis, Y. Eren, and J. P. Catalão, "End-user comfort oriented day-ahead planning for responsive residential HVAC demand aggregation considering weather forecasts," *IEEE Trans. Smart Grid*, vol. 8, pp. 362-372, Jan. 2017.
- [112]X. Wu, J. He, Y. Xu, J. Lu, N. Lu, and X. Wang, (2018). "Hierarchical control of residential HVAC units for primary frequency regulation," *IEEE Trans. Smart Grid*, vol. 9, pp. 3844-3856, Jul. 2018.
- [113]Y. Lin, P. Barooah, and J. L. Mathieu, "Ancillary services through demand scheduling and control of commercial buildings," *IEEE Trans. Power Syst.*, vol. 32, pp. 186-197, Jan. 2017.
- [114]L. Yu, T. Jiang, and Y. Zou, "Online Energy Management for a Sustainable Smart Home with an HVAC Load and Random Occupancy," *IEEE Trans. Smart Grid*, vol. 10, pp. 1646-1659, Mar. 2019.
- [115]H. Hao, C. D. Corbin, K. Kalsi, and R. G. Pratt, "Transactive control of commercial buildings for demand response," *IEEE Trans. Power Syst.*, vol. 32, pp. 774-783, Jan. 2017.

- [116]H. Sun, F. Zhao, H. Wang, K. Wang, W. Jiang, Q. Guo, B. Zhang, L. Wehenkel, “Automatic learning of fine operating rules for online power system security control,” *IEEE Trans Neural Networks and Learning Systems*, vol. 27, pp. 1708-1719, Aug. 2016.
- [117]V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” arXiv preprint arXiv:1312.5602.
- [118]Y. Wu, H. Tan, J. Peng, H. Zhang, H. He, “Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus,” *Applied Energy*, vol. 247, pp. 454-466, 2019.
- [119]X. Han, H. He, J. Wu, J. Peng, Y. Li, “Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle,” *Applied Energy*, vol. 254, pp. 113708, 2019.
- [120]H. Hua, Y. Qin, C. Hao, J. Cao, “Optimal energy management strategies for energy Internet via deep reinforcement learning approach,” *Applied Energy*, vol. 239, pp. 598-609, 2019.
- [121]R. Rocchetta, L. Bellani, M. Compare, E. Zio, E. Patelli, “A reinforcement learning framework for optimal operation and maintenance of power grids,” *Applied Energy*, vol. 241, pp. 291-301, 2019.
- [122]P. Kou, D. Liang, C. Wang, Z. Wu, L. Gao, “Safe deep reinforcement learning-based constrained optimal control scheme for active distribution networks,” *Applied Energy*, vol. 264, pp. 114772, 2020.
- [123]T. Wei, S. Ren, and Q. Zhu, “Deep Reinforcement Learning for Joint Datacenter and HVAC Load Control in Distributed Mixed-Use Buildings,” *IEEE Trans. Sustainable Computing*, early access, 2019.
- [124]Y. Wang, K. Velswamy, and B. Huang, “A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems,” *Processes*, vol. 5, no. 3, Aug. 2017.
- [125]G. Gao, J. Li, J., and Y. Wen, “Energy-Efficient Thermal Comfort Control in Smart Buildings via Deep Reinforcement Learning,” arXiv preprint arXiv:1901.04693, 2019.
- [126]N. Lu, “An Evaluation of the HVAC Load Potential for Providing Load Balancing Service,” *IEEE Trans. Smart Grid*, vol. 3, pp. 1263-1270, Sep. 2012.
- [127]B. Cui, J. Joe, J. Munk, J. Sun, T. Kuruganti, “Load Flexibility Analysis of Residential HVAC and Water Heating and Commercial Refrigeration,” Oak Ridge National Lab, Oak Ridge, TN (United States), Sep. 2019.
- [128]V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. B. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529-533, 2015.
- [129]T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al. “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.

- [130]B. Cui, J. Munk, R. Jackson, D. Fugate, and M. Starke, "Building thermal model development of typical house in U.S. for virtual storage control of aggregated building loads based on limited available information," in 30th International Conference on Efficiency, Cost, Optimization, Simulation, and Environmental Impact of Energy Systems, 2017.
- [131]Clean Power Research. [online]: <https://www.cleanpower.com/>
- [132]TensorFlow. [online]: <https://www.tensorflow.org/>
- [133]PJM market. [online]: <https://www.pjm.com/>
- [134]M. Shafie-khah, and J. P. Catalão, "A stochastic multi-layer agent-based model to study electricity market participants behavior," *IEEE Trans. Power Syst.*, vol. 30, pp. 867-881, Mar. 2015.
- [135]I. Taheri, M. Rashidinejad, A. Badri, and A. Rahimi-Kian, "Analytical approach in computing Nash equilibrium for oligopolistic competition of transmission-constrained GENCOs," *IEEE systems journal*, vol. 9, pp. 1452-1462, Dec. 2015.
- [136]B. Zhang, R. Johari, and R. Rajagopal, "Competition and coalition formation of renewable power producers," *IEEE Trans. Power Syst.*, vol. 30, pp. 1624-1632, May 2015.
- [137]L. Du, S. Grijalva, and R. G. Harley, "Game-Theoretic Formulation of Power Dispatch with Guaranteed Convergence and Prioritized Best Response," *IEEE Trans. Sustain. Energy*, vol. 6, pp. 51-59, Jan. 2015.
- [138]M. Kazemi, H. Zareipour, F. Ehsan, and W. D. Rosehart, "A robust linear approach for offering strategy of a hybrid electric energy company," *IEEE Trans. Power Syst.*, vol. 32, pp. 1949-1959, May 2017.
- [139]G. R. Gajjar, S. A. Khaparde, P. Nagaraju, and S. A. Soman, "Application of actor-critic learning algorithm for optimal bidding problem of a Genco," *IEEE Trans. Power Syst.*, vol. 18, pp. 11-18, Feb. 2003.
- [140]H. Kebriaei, A. Rahimi-Kian, and M. N. Ahmadabadi, "Model-based and learning-based decision making in incomplete information Cournot games: A state estimation approach," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, vol. 45, pp. 713-718, Apr. 2015.
- [141]M. R. Salehizadeh, and S. Soltaniyan, "Application of fuzzy Q-learning for electricity market modeling by considering renewable power penetration," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 1172-1181, Apr. 2016.
- [142]D. Esmaili Aliabadi, M. Kaya, and G. Sahin, "Competition, risk and learning in electricity markets: An agent-based simulation study," *Applied Energy*, vol. 195, pp. 1000-1011, 2017.
- [143]N. Rashedi, M. A. Tajeddini, H. Kebriaei, "Markov game approach for multi-agent competitive bidding strategies in electricity market," *IET Generation, Transmission & Distribution*, vol. 10, no. 15, pp. 3756-3763, Nov. 2016.

- [144]Y. Ye, D. Qiu, M. Sun, D. Papadaskalopoulos, and G. Strbac,“Deep Reinforcement Learning for Strategic Bidding in Electricity Markets,” *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1343-1355, Mar. 2020.
- [145]Y. Ye, D. Qiu, J. Li, and G. Strbac, “Multi-Period and Multi-Spatial Equilibrium Analysis in Imperfect Electricity Markets: A Novel Multi-Agent Deep Reinforcement Learning Approach,” *IEEE Access*, vol. 7, pp. 130515-130529, Sep. 2019.
- [146]Y. Huang, J. Shang, C. Kang, et al. “An Operation Mechanism and Model of the Day-ahead Electricity Market”, *Automation of Electric Power Systems*, vol. 27, no. 3, pp. 23-27, Feb. 2003.
- [147]R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Advances in Neural Information Processing Systems (NIPS 2017)*, 2017, pp. 6379-6390.

Appendix

Publications During Ph.D. Study

Journal papers

- [1] F. Li and **Y. Du**, “From AlphaGo to Power System AI Applications,” *IEEE Power & Energy Magazine*, vol. 16, pp. 76-84, Feb. 2018.
- [2] **Y. Du**, Z. Wang, G. Liu, X. Chen, H. Yuan, Y. Wei, and F. Li, “A Cooperative Game Approach for Coordinating Multi-Microgrid Operation within Distribution Systems,” *Applied Energy*, vol. 222, pp. 383-395, Jul. 2018.
- [3] **Y. Du**, F. Li, J. Li, and T. Zheng, “Achieving 100x Acceleration for N-1 Contingency Screening with Uncertain Scenarios using Deep Convolutional Neural Network,” *IEEE Trans. Power Systems*, vol. 34, pp. 3303-3305, Jul. 2019.
- [4] **Y. Du** and F. Li, “A Hierarchical Real-time Balancing Market Considering Multi-microgrids with Distributed Sustainable Resources,” *IEEE Trans. Sustainable Energy*, vol. 11, pp. 72-83, Jan. 2020.
- [5] **Y. Du** and F. Li, “Multi-microgrid Energy Management based on Deep Neural Network and Model-free Reinforcement Learning,” *IEEE Trans. Smart Grid*, vol. 11, pp. 1066-1076, Mar. 2020.
- [6] **Y. Du**, F. Li, T. Zheng, and J. Li, “Fast Cascading Outage Screening based on Deep Convolutional Neural Network and Depth-First Search,” *IEEE Trans. Power Systems*, 2020, early access.

Conference papers

- [1] **Y. Du**, F. Li, X. Kou, and W. Pei, “Coordinating Multi-microgrid Operation within Distribution System: A Cooperative Game Approach,” in *2017 Power & Energy Society General Meeting*, pp. 1-5.
- [2] **Y. Du** and F. Li, “Integrating a Multi-microgrid System into Real-time Balancing Market: Problem Formulation and Solution Technique,” in *2018 Power & Energy Society General Meeting*, pp. 1-5.
- [3] **Y. Du**, F. Li, H. Yuan, and Y. Wei, “Distribution Locational Marginal Price for Grid-connected Microgrids in Real-time Balancing Market”, in *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*, pp. 1-5.
- [4] **Y. Du**, F. Li, and C. Huang, “Applying Deep Convolutional Neural Network for Fast Security

Assessment with N-1 Contingency,” in *2019 Power & Energy Society General Meeting*, pp. 1-5.

- [5] **Y. Du**, J. Zhao, T. Zheng, E. Litvinov, F. Li, “Multi-Day-ahead Net Interchange Schedule Forecast based on LSTM Recurrent Neural Network”, accepted by *2020 Power & Energy Society General Meeting*.

Vita

Yan Du received her B.S. degree from Tianjin University, Tianjin, China, in 2013, and M.Sc. degree from Institute of Electrical Engineering, Chinese Academy of Sciences, Beijing, China, in 2016, both in Electrical Engineering. She is currently pursuing her Ph.D. degree at the University of Tennessee at Knoxville under the guidance of Professor Fangxing Li. Her research interests include electricity market, optimization, and deep learning applications in power systems.