Lakehead University

Peak Load Ensemble Prediction and Multi-agent Reinforcement Learning for DER Demand Response Management in Smart Grids

by

Jiawei Dong

A thesis submitted in partial fulfillment for the degree of Master

in the
Engineering of Electrical & Computer

April 2022

Declaration of Authorship

Lakehead University

Abstract

Faculty Name
Engineering of Electrical & Computer

Master of Engineering

by Jiawei Dong

The increasing number of Distributed Energy Resources (DERs), such as home batteries and Electrical Vehicles (EVs), provides an opportunity for utility companies to develop demand response mechanisms to balance the demand and supply of energy during peak times. However, it is challenging to shave the grid's peak load efficiently and effectively as it requires accurate energy forecasting and coordinated management of DERs. To address this challenge, this thesis proposes a system consisting of an image-based ensemble prediction model and a Multi-agent Reinforcement Learning (MARL) mechanism for Demand Response (DR) management in smart grids. For the image-based prediction model, we hypothesize that the approximate curve of the daily power consumption graph has some specific patterns that can be used to separate each day into different groups based on the pattern of the energy consumption curve. To this end, we use a convolution neural network model to classify and extract the features of the curve image. Then, we apply the k-means mechanism for image clustering to select better training sets and optimize the forecasting mechanism. Our results show an overall improvement in prediction during the season-changing period. The proposed MARL mechanism takes the prediction results as input to the agents to coordinate the discharging time of DERs to maximize the peak shaving performance. This mechanism requires centralized training and allows distributed execution. The system's evaluations and experiments are conducted on a real-life dataset, and our results show the proposed system's effectiveness.

Acknowledgements

I am appreciative of the support of my supervisor, Dr. Abdulsalam Yassine. This thesis and research would not have been possible without his exceptional support. I am grateful for his constructive suggestions, patient instruction, and enthusiastic encouragement, especially during the pandemic.

I thank Lakehead University and the Department of Electrical and Computer Engineering for providing scholarships and awards as financial support. I am also thankful for the lab space and online remote desktop services, which help improve my productivity.

I am grateful to Synergy North, the company that cooperates with us on this project. They provide the dataset of energy consumption, which is very valuable during my research. I am grateful for Andy Armitage, the Vice President, Customer and Information Services at Synergy North, who offers instruction on dataset connection and professional advice for our project.

I am also grateful to my lab mates. Their welcoming gives me a wonderful research time. Our discussion about the newest techniques and their research directions always provide me with new insight.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
CNN	**C**onvolution **N**eural **N**etwork
DER	**D**istributed **E**nergy **R**esource
DL	**D**eep **L**earning
DR	**D**emand **R**espond
EV	**E**lectric **V**ehicle
FNN	**F**eedfoward **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
MARL	**M**ulti **A**gent **R**einforcement **L**earning
MARIMA	**M**ultivariate **A**utoregressive **I**ntegrated **M**oving **A**verage
MDP	**M**arkov **D**ecision **P** rocess
NN	**N**eural **N**etwork
RL	**R**einforcement **L**earning
RMSE	**R**oot **M**ean **S**quare **E**rror
SGP	**S**mart **G**rid **P**rovider
V2G	**V**ehicle **G**rid

Symbols

$\mathcal{N}$	Set of agents
i	Index of an agent
c_i	The predefined battery discharging capacity of agent battery i in kWh
k_i	The predefined threshold of discharging an of agent battery i in kWh
d	Index of day
t	Index of time
x	The number of discharging duration
p_i	Discharging rate of agent battery i in kW
ep_d^t	Predicted energy power consumption in kW for day d at time t
EP^d	Predicted energy power consumption in kW for day d for 24 hours period
λ_l	The off-peak charging price \$/kWh
λ_h	The on-peak charging price \$/kWh
τ_d^i	The discharging period of agent i in day d
S	Finite state space
A	Action space
r_i	Reward of the agent i
R_d	Shared Reward of the agents of day d
π	Policy function
θ	Trainable parameters for policy neural network
w	Trainable parameters for value neural network
α	Learning rate for value neural network
β	Learning rate for policy neural network
ε	Exploration decay
q/Q	Action value function
V	State value function

J	Objective function
L	Loss function
g	derivative of the state value function

I dedicate my work to my parents, Yuliang Dong and Jianping Zou. Not only for their financial support but also for their love and encouragement to me. Even though they live in my hometown China, their support is the biggest reason I can focus on my research without other worries.

Chapter 1

Introduction

1.1 Introduction

Energy management systems that deal with the integration of DERs into the grid system require an accurate prediction model and a sophisticated coordination mechanism to balance the demand and supply. For DERs that consist of smart home battery systems or EV batteries, load forecasting becomes a crucial component because the energy management system demands a short-term peak load prediction to engage loads from such DERs. This is because these battery systems are relatively small; thus, they require shorter charging and discharging time [1],[2], and [3]. Thus, an accurate peak load prediction model has a significant impact on developing an appropriate mitigation strategy toward load balancing [4] and resource planning effectiveness [5]. To address this challenge, this thesis proposes a system consisting of an image-based ensemble prediction model and a Multi-agent Reinforcement Learning (MARL) mechanism for DR management in smart grids. For the image-based prediction model, we hypothesize that the approximate curve of the daily power consumption graph has some specific patterns that can be used to separate each day into different groups based on the pattern of the energy consumption curve. To this end, we use a convolution neural network model to classify and extract the features of the curve image. Then, we apply the k-means mechanism for image clustering to select better training sets and optimize the forecasting mechanism. Our results show an overall improvement of prediction during the season-changing period. The proposed MARL mechanism takes the prediction results as input to the agent to coordinate the discharging time of DERs to maximize the peak shaving performance. This mechanism requires centralized training and allows distributed execution.

The literature has a plethora of studies on energy forecasting models. For example, the work in [6] compares different algorithms of energy forecasting in smart cities while the work in [7] focuses on individual house energy data. After analyzing these studies, we found that the decreased size of the geographically collected energy data leads to increased randomness; thus, it is harder to perform accurate forecasting. The work presented in [8] addresses the problem of prediction accuracy of multivariate models. The study proposes a hybrid system to analyze the energy consumption data along with associated weather data at different time periods and address the limitations of learning techniques. The model combines both the long-term and the short-term learning mechanisms to improve performance and accuracy. In [9], the authors propose a model that allows utility companies to predict the highest peak of energy consumption on the distribution feed where the load is connected. The prediction model in the study uses Multivariate Autoregressive Integrated Moving Average (MARIMA) to reach an accuracy of 92.64%. Parallel to the above-mentioned studies, the work described in [10] looks at the season-changing period, where energy consumption activities are more random because of weather changes. The decrease in accuracy during the season-changing period is related to over-fitting from unexpected data. In general, season-changing does not have a specific time throughout the year, so we can not use the season as a variable in the training set. However, the season is an essential factor that affects power consumption. To improve the model accuracy in the shoulder season, we hypothesize that the approximate curve of the daily power consumption graph has some specific patterns. We can separate each day into different types based on the pattern of the energy consumption curve image, where the data in the same type are primarily correlated. Although the consumption value in kWh of each day is different, the shape of the curve almost exhibits a similar pattern for each day. As a result, we use image processing to separate similar patterns of power consumption curves to different groups. Then, we implemented an ensemble learning model with different learners. The ensemble learning combines multiple weakly supervised models to obtain an outstanding strong-supervised model. The idea is that different learner's correct errors mutually to achieve the ultimate accuracy improvement.

As mentioned above, the main purpose of the energy management system is to balance the demand and supply of the energy grid. One method of achieving the balance is through the coordination of DERs to shave the peak load. But, before shaving the peak, we need to predict when it will happen. This is usually done in the day ahead. Based on the optimized day-ahead power consumption prediction, a coordination model is used to integrate the DERs to shave the peak. This thesis proposes a MARL model with the coordination of DERs to shave the peak. Reinforcement learning (RL) has recently received significant attention from researchers as an effective AI-based technique to coordinate the integration of DERs (e.g., EVs, battery storage, etc.) into the smart grid for an efficient balance of supply and demand [11],[12],[13], and [14]. In RL schemes, DERs are modelled as agents who interact with the environment (e.g., demand response programs, utility companies, price signals, etc.) to take actions and receive rewards (e.g., monetary benefit, cost reduction, etc.). In such a process, the agents learn from their actions and observe the outputs and rewards continuously. The agents can be trained to be autonomous, adaptive, independent of the environment model, which makes them practical to DR applications [13][14]. In smart grids, DR programs are expected to engage a variety of DERs to reduce the peak load; hence, it is required that the DR system relies on a scheme that does not require full observability of the entire system to make decisions. Indeed, it is not practical to assume that DERs of different types and brands will be interacting with each other to coordinate the dispatch process in DR programs [13]. Agents in RL can make decisions with partial observability of the entire system. Several researchers have attempted to solve the DER coordination problem using RL. However, most of these studies focus on the uncertainty of the environment [15], the scaling of the system by increasing the number of agents [16], and the comfort of the users [17]. This work is completely different from previous attempts because we tackle the day-ahead coordination of the DERs to reduce the peak load. Our system depends on an external prediction mechanism to determine the day-ahead peak period. However, it provides discharging scheduling of DERs with an optimized peak shaving performance while considering the inaccuracy of energy prediction. The model depicts DERs as autonomous agents and provides centralized training and distributed execution. In the proposed model, DERs scheduling does not need to constantly connect to a centralized server in the distributed execution phase, which improves the system's robustness and provides data privacy for DER owners. This is also a significant distinction from existing studies.

1.2 Technical Challenges

In the energy arbitrage with DERs, the day-ahead forecasting accuracy is significant due to the short discharging period of the DERs. If the discharging period did not cover the highest peak period, then the peak load still exists. Thus, an accurate peak load prediction model has a significant impact on load balancing [4] and resource planning effectiveness. Energy forecasting in a small residential area has more randomness than in a larger residential area or industry area. Unlike industry areas, the energy consumption of residential areas is highly related to people's activities. People in residential areas use the energy more randomly, and the smaller the area size is, the more difficult it is to predict. Besides, the weather is an essential factor in forecasting peak load. The correlation between weather and power consumption changes over the season, making energy forecasting during shoulder season (season-changing period) more difficult. The nature of weather change is brutal to detect the exact season; thus, using the season as a parameter is hardly performed.

One the energy consumption is forecasted, managing the DERs with efficient day-ahead discharging scheduling is another challenge. Assume we have accurate day-ahead forecasting; scheduling every DER at the highest peak is inefficient management. The highest peak is shaved but over-clipped and create two relative peaks at two sides of the original peak period. The discharging of one EV battery affects other discharging consequences on peak-shaving. When several batteries discharge at the same peak load time slot, the highest peak of the load is over clipped into a relative valley and creates another two relative peaks instead of effectively flat the peak load. Figure 1.1-a describes the peak shaving in which the discharging process is creating two relative peaks and figure 1.1-b displays the proposed solution. The difference between the two figures clearly shows the need for a coordinated discharging process to flatten the peak effectively. It is challenging to avoid DERs discharge at the same peak load time slot and maximize the efficiency of flattening the peak load. Besides, daily consumption may have two peak periods with similar power consumption or a flatter and longer peak period. Ideally, DERs should be coordinated to deal with different types of daily consumption peak curves and flatten power consumption curves. Different DERs belong to different owners, and owners may not be willing or able to share their scheduling information, making coordination between DERs difficult.

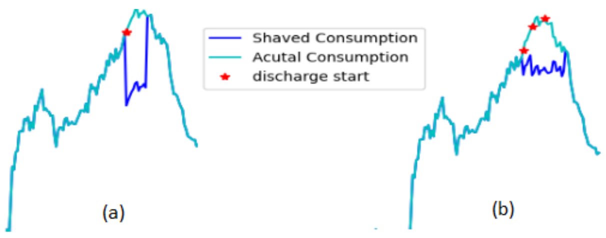


FIGURE 1.1: (a) Reducing highest peak and creating to 2 relative peaks, (b) Flattening the demand peak effectively

1.3 Research Approach

In the challenge of optimizing the power consumption prediction, we focus on improving model performance during shoulder seasons. We hypothesize that the daily power consumption curve images have specific patterns and data of same type days are primarily correlated. To overcome the difficulty of clustering the daily power consumption curve, our approach uses a convolution neural network to extract features and use the k-means mechanism for clustering. We then select the training set based on the clustered type of each day. To further increase the prediction accuracy, we also propose an ensemble model combining multiple algorithms. The prediction results averaging through a dynamic weighting system that outputs a predicted curve and a majority voting mechanism to determine the peak point's output.

Driven by the challenge of cooperatively discharge scheduling of DERs, we propose a MARL mechanism that provides a day-ahead discharging schedule for each DER. Our approach is different from existing studies as it focuses on the peak shaving efficiency of scheduling discharging DER in the local grid. In this approach, each battery agent receives day-ahead overall energy consumption forecasting as mentioned above and produces day-ahead discharging scheduling based on the agent's properties and policy. The proposed MARL models are centralized training and distributed execution to provide day-ahead optimized DER discharging scheduling of DER. The policy of agents is trained by a neural network model of an actor-critic algorithm [18]. The model contains two neural networks: the critic network estimates the reward of each agent's discharging schedule (action-value). The actor-network updates the policy based on the suggestion of the critic network. This advanced technology improves our model's complexity, increasing the learning performance and providing a potential for future distributed training. The model offers a corresponding scheduling strategy based on the confident interval of daily forecasting learned during the simulated training process. The agents have a cooperative relationship, which allows them to provide coordinated scheduling to maximize daily peak shaving performance in the distributed execution without communication after training. Local agents' distributed execution enhances network connection robustness and improves general DER owners' data privacy from presenting their daily discharging plan collected in a centralized server.

1.4 Contribution

The following are the contributions of the thesis.

1. We hypothesize that the power consumption primarily correlated to the previous days in the same type of curve as the predicted day. In the experiments, we prove that the selection of training sets containing the same type as the predicted days will increase the accuracy of the models, especially during shoulder season. The results of experiments prove that, in the residential area, the daily power consumption curve has certain types, and days in the same type have a stronger correlation. We can increase the model performance by clustering these types. To achieve the above tasks, we propose a novel image-based processing technique to cluster the daily power consumption curve based on the load curve's characteristics. We used a pre-trained CNN to extract the features of the image and use the

k-means mechanism for clustering. The proposed technique solves the difficulty of deciding the number of clusters of the training set. Transferring the knowledge in the pre-trained CNN model tackles the shortage of labelled daily power consumption curve images. The image-based processing technique converts the task of clustering the daily power consumption curve image from an unsupervised problem to a supervised one. Also, we propose an ensemble day-ahead load forecasting model with dynamic weighting and a majority voting mechanism. The proposed model's ensemble multiple machine learning algorithm includes Random Forest, XGBoost, Cubist, Feedforward Neural Network and Long-Short Term Memory. The experiments proved that the proposed ensemble models have improved performance in predicting accuracy factors such as Root Mean Square Error (RMSE) and the highest peak difference.

2. For the coordination of the DERs' charging/discharing process, we propose a robust MARL model to coordinate the integration of DERs into the smart grid system. Particularly, we showed that our model effectively reduces the peak load regardless of the inaccurate energy forecasting days. This is rather significant because it helps utility companies to minimize their peak load cost and prevent the creation of relative peaks. The proposed MARL model provides a centralized training distributed execution that allowed DERs to coordinate the peak-shaving tasks without information sharing or communication between the agents. The importance of this approach is that it is practical and encourages agents to participate in the system because the distributed execution does not need a collection of discharging times of DERs in a centralized server. Therefore, data privacy is maintained, and customers' interest in enrolling in energy arbitrage can increase.

1.5 Publications

J. Dong, A. Yassine and A. Armitage, "Image-based with Peak Load Ensemble Prediction System for Demand Response in Smart Grid," 2021 International Symposium on Networks, Computers and Communications (ISNCC), 2021, pp. 1-6, doi: 10.1109/ISNCC52172.2021.9615837.

J. Dong, A. Yassine and A. Armitage, "Image-Based Processing Mechanism for Peak Load Forecasting in Smart Grids," 2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE), 2020, pp. 64-69, doi: 10.1109/SEGE49949.2020.9182025.

1.6 Organization

This thesis is organized as follows:

- Chapter 1: Introduction -This chapter provides a high-level description of the thesis and all aspects involved.
- Chapter 2: Background, Related Work, and overall architecture of the proposed model - This chapter describes the fundamental idea of energy arbitrage, short-term energy forecasting and reinforcement learning. This chapter also presents a summary of relevant published works. The works principally include research on parameters correlation analysis, standard models of short-term energy forecasting, and DER management application with MARL.
- Chapter 3: Novel Image-based Peak Load Ensemble Prediction System - This chapter proposes a novel image-based ensemble system that enhances prediction outcomes, especially during the shoulder season (season-changing days).
- Chapter 4: MARL Approach to Day-ahead DER Discharging Scheduling - This chapter proposed an optimal day-ahead peak-shaving scheduling system based on the MARL approach on energy forecasting of DERs.
- Chapter 5: Conclusion and Future Work - This chapter describes the conclusions from the analysis of the results of this work. We propose some possible future extensions and a review of the current shortage.

Chapter 2

Background, Related Work and Proposed Model

In this chapter, we introduce the background and related work including studies on short-term energy forecasting in residential areas and MARL in DER management. We also present the overall architecture of the proposed model.

2.1 Energy Arbitrage and Forecasting Models

Peak shaving and valley filling is a measure to flatten the electricity load during the day. Such a process requires managing and organizing the electricity consumption time of various users in coordinated manner. The aim is to reduce the peak-to-valley difference of the grid load so that the power generation and electricity consumption become balanced. Because power plants continue to generate electricity around the clock, if the electricity generated is not used, the excess energy from power generation will be wasted. The power generation capacity of a power plant is usually fixed and does not change easily. However, the peak of electricity consumption happen during the day when there is insufficient generation of electricity. But, the electricity usage is low at night when excess of electricity is not used. In response to this phenomenon, power companies shift part of the peak load to the period at night, thereby using the excess power at night and achieving the goal of energy balance. However, to achieve this shift, they need to predict the expected peak times with high accuracy. Energy forecasting is an important part of power system planning, and it is also the foundation of power system economic operation. Energy forecasting refers to the research or use of mathematical methods to systematically process past and future loads under the conditions of fully considering some important system operating characteristics.

This thesis is concerned with day-ahead, or short-term load forecasting, which is affected by various factors such as weather changes, social activities and festival types and appears as a non-stationary random process in the time series [51]. However, most of the factors that affect the system load are regular. There are many methods used for short-term load forecasting. The newer algorithms mainly include the neural network, time series, regression analysis, support vector machine, fuzzy forecasting [4]. The core challenge of power load forecasting research is using the existing historical data to establish a forecasting model to predict the load value in the future, such as the work in [23]. Therefore, the reliability of historical data information and the forecasting model are two main factors affecting short-term load forecasting accuracy. With the gradual establishment of the database of power systems and the improvement of weather forecasting, it is no longer difficult to obtain various historical data accurately. Therefore, the core issue of short-term load forecasting is the mathematical forecasting models.

Other studies focus on machine learning to produce advanced load forecasting methods. The application of the neural network methods in load forecasting is mainly divided into an Artificial Neural Network (ANN) such as [4] and a Recurrent Neural Network (RNN) [37]. The advantages of the neural networks model are that it can adapt to a large number of non-structural and imprecise rules. The neural network also has strong calculation and complex mapping capabilities as well as fault tolerance. The shortcomings of the neural network method include difficulty in the construction of the model structure, optimization of the learning speed, and the local minimum problem.

Several forecasting methods use historical data of power load, which is an ordered collection sampled and recorded at a specific time interval, so it is a time series. The time series methods in [7][46], and [47] are relatively mature algorithms for short-term load forecasting of the power system. The advantage of the time series methods is that they require less data, less workload, faster calculation speed, and reflect the continuity of recent changes in load. The disadvantage of the time series methods is that the modelling process is more complicated than other methods. The model has high requirements for the stability of the original time series and is only suitable for short-term forecasts with relatively constant power consumption changes. The time series methods do not consider other influences of load changes. For example, when the weather changes significantly or during holidays, the model's prediction error is relatively large.

Existing work related to regression analysis forecasting methods such as [44] and [45] finds the correlation between the independent and dependent variables and its regression equation according to the changing rules of historical power consumption data and the factors that affect the load change. These methods determine the model parameters and infer the load value in the future. The advantages of regression analysis are simple calculation principle and structure, fast prediction speed, good extrapolation performance, and better predictions for situations that have not appeared in history. The shortcomings of the regression method are that it requires lots of historical data. Furthermore, because of regression method uses linear methods to describe a complex problem such as energy consumption, the structure is too simple, and the accuracy is low. The model cannot describe various factors that affect the load, and the model initialization is challenging.

2.2 Reinforcement Learning - a Brief Background

RL is an area of machine learning method for computers (Agent) that learns the optimal strategy (Policy) by continuously interacting with the possibly unknown environment for a given task. In a particular task state, the traditional search strategy needs to search down layer by layer. It takes much time to calculate the optimal operation in this state. RL can directly learn a mapping from state to operate. This mapping can be a function or a lookup table that stores states and operations [56]. Therefore, RL can always get the optimal action faster in real-time tasks and save more memory. In addition, it is often not clear how the task environment interacts with the operation. At this time, the algorithm needs actual exploration in a simulated or actual environment. We can only rely on RL's adventurous properties, which constantly update information in interactions. RL can quickly adapt through learning if the environment changes instead of manually remodelling and remodifying the environment like traditional searching. In this thesis, RL is applied to the design of the day-ahead DER scheduling system because of the above mentioned characteristics.

The four most basic elements in RL are the following [61]: State (S); there can be many states in a task, and we set each state equidistant in time. Action (A): There should be at least one action for each state to choose. Reward (R); the environment will provide numerical feedback on the selected action for each state. The higher the value, the more favoured the action. Policy (π); given a state as input, policy π always output only one operation a, that is, $a = \pi(s)$, π can be a lookup table or a function.In general, the thinking process of RL is: given a task, the agent will get a state every fixed time step. The agent needs to select an action from the preset action list and implement it. In the next time step, the agent will get the state of the next time step and the feedback of it. Clearly, each time step the agent does two things: get the state of the time step and the feedback of the previous step; give the operation of the step from policy and interact with the environment.

The agent will also update the policy based on the information obtained to learn the optimal strategy [62]. The agent's policy is updated during interaction with the environment (online update) or after the end of the episode (offline update). The agent also could be updated online and offline at the same time. All operations of the agents are only based on the information obtained in each time step. In other words, the agent only needs the current time step's state information to give a proper action. It has nothing to do with the information before this step. The characteristics considering future development is only related to the present is called Markov. The time series composed of states with this characteristic is called Markov Process. When decisions are invlived in each state, the whole process is the Markov Decision Process (MDP). Most of the current RL is based on the idea of MDP.

Finding the optimal policy is the core objective of RL. Before considering how to find the optimal strategy, we need a way to evaluate each policy. Suppose that an agent is designed to play chess, and the chess played by the agent is defined as an action. The initial position or each position after the opponent's move is a state until the checkmate becomes the opponent's victory; otherwise, the agent loses. If there are two policies π and π^1, they both win. Can it be said that the two strategies are equally good? We can define it this way, but experience tells us that not every operation in a task is often optimal. The same strategy will likely have different results in different chess tasks, so it is impossible to compare the advantages and disadvantages of different strategies.

2.2.1 Multi-agent Reinforcement Learning

The practicality and scalability of a multi-intelligent system allow it to be employed in multiple fields such as robot cooperation, distributed control, resource management, collaborative decision support systems, autonomous fighting systems, and data mining. In multi-agent systems, the agents interact with the environment simultaneously [76]. Each agent still follows the goal of reinforcement learning, which is to maximize the cumulative return obtained. The change in the state of the environment is related to the actions of all agents; therefore, the agent's strategy learning will consider the influence of joint actions. The expansion of the MDP to multi-agent systems requires the knowledge of game theory to model multi-agent reinforcement learning problems [75]. Compared with a single-agent system, the application of RL in a multi-agent system will encounter the following problems and challenges:

- Environmental instability: While the agent is making decisions, other agents are also taking action. Changes in the state of the environment are related to the joint actions of all agents. The return of the same agent taking the same action in the same state may differ caused by the change in other agents' actions.

- Limitation of information acquisition: Agents may only obtain partial, instead of global, observational information. Agents cannot know other agents' observational information, actions, and rewards.
- Consistency of goal: The objective of each agent may be the best global return or may also be the best individual return. The design of the agent's objective depends on the real problem.
- Scalability: In a large-scale multi-agent system, high-dimensional state space and action space are involved. There are specific requirements for model expression capabilities and hardware computing power in real scenarios.

The algorithms address how an intelligent agent executes actions in the environment to maximize the cumulative reward, in contrast to the aims of supervised and unsupervised learning. Reinforcement learning models can be used to abstract a wide range of control and decision problems. Like supervised learning, reinforcement learning includes a training process that entails repeatedly performing actions, observing the consequence of those actions, and gathering experience to construct a model [74]. In contrast to supervised learning, there is usually no direct calibrated label value for each action as a supervisory signal. The system only provides feedback to the algorithm's action, which is usually delayed. The consequences of the current action will be fully reflected in the future with randomness.

The literature discusses several solutions/algorithms [79] [80], and [81] such as dynamic programming, which can theoretically solve the reinforcement learning problem if the state transfer probabilities of all states and the payoff values are known. Starting with a random initial value, some rule is employed to iterate until the state-value function or action-value function converges to an extreme value. The Bellman equation is commonly used as an iteration rule, also known as the Bellman optimality equation [78]. In many actual situations, however, we cannot obtain the transfer probabilities of all states and must instead rely on stochastic techniques. The basic idea is to undertake a series of random actions, starting with a random strategy, and then observe the payoffs and state transfers to estimate or update the value of the value function. It means increasing the execution of influential acts while decreasing the execution of ineffective actions. Typical examples are the Monte Carlo algorithm and the temporal difference algorithm. After an action is completed, the temporal difference algorithm [57] updates the action-value function. The TD approach does not need to use the state transfer probability and instead calculates it directly using random samples. Using the Bellman equation, the TD algorithm evaluates the value of the value function and then creates the update term. The SARSA algorithm and the Q-learning method are two popular implementations. The Q-learning method [58] is a type of temporal difference algorithm that predicts the maximum value of the value function for each action. The extreme value of the Q-function can be obtained directly by iteration to find the ideal policy.

The following is the process of the Q-learning training algorithm: initialization [58]: for all non-terminating states, set $Q(s, a)$ to any value, and for terminating states, set $Q(s, a)$ to 0. The implementation necessitates choosing one of three actions for each state to conduct based on the current estimate of the action-value function. The first is to choose an action at random, known as exploration, and the second is to choose the action with the highest value based on the current action function, which is known as exploitation. The third technique, known as the $\varepsilon - greedy$ strategy, is a hybrid of the preceding two. After completing an action, it enters the next state s' and generates an update term by finding the extreme value of the value function of all actions in state s'. The method finally converges on the action value function's best value. When used for prediction, the action with the highest function value in each state is chosen for execution, resulting in the best strategy. The same $\varepsilon - greedy$ strategy can be utilized for the specific implementation. For a specific implementation, all states and actions are recorded in Q(s, a) in a two-dimensional table that is first initialized, then utilized to determine which actions should be conducted. Finally, the table's values are updated until convergence.

Deep RL (DRL) that integrates deep neural networks with RL has become a prevalent study issue for all researchers due to the advent of deep learning technology and its outstanding results in many sectors. They have actively attempted to merge DRL methodologies into the multi-agent system to achieve various complex tasks in multi-agent environments. Multi-agent Deep Reinforcement Learning (MDRL) [74], widely employed in several real-world areas after several years of development and innovation, has evolved into multiple algorithms, rules, and frameworks. MDRL is becoming the hottest study and application direction in machine learning. MARL is based on the Stochastic Game (SG) [75] method, as opposed to single-agent RL. MARL algorithms can be characterized as fully cooperative, fully competitive, or mixed [76].

DQN (Deep Q Network), introduced by DeepMind in 2013 [66] and improved in Nature in 2015, is a typical example of DRL based on value functions. This method uses a CNN to fit a value function, usually a Q function. The original scene data, such as a gaming screen image, is fed into the network. The output is the extreme value of the Q function achieved by performing various actions in that scenario. Despite DQN's success, there is still much opportunity for improvement. Since then, many better algorithms have appeared for DQN, including enhancements to the general system structure, training sample construction, and neural network structure. In the literature, the Double DQN (DDQN) algorithm was proposed [68]. In the DDQN, there are two sets of parameters: θ and $\theta-$. Theta is used to choose the action with the highest Q value; θ and $\theta-$ are used to determine the ideal action's Q value. By separating action selection and policy evaluation with these two sets of parameters, the possibility of overestimating Q is reduced. DDQN selects the ideal action using the current value network's parameter θ and evaluates that action using the target value network's parameter $\theta-$. The experimental results show that DDQN can more reliably estimate the Q function value, resulting in a more stable training process and trained policy.

A priority sampling-based DQN, which enhances the empirical replay mechanism, is proposed in [69]. In prior DQNs, the training samples were randomly sampled with equal probability from the samples in the experience pool at each iteration. The prior DQNs did not take into account the significance of each sample. The method described in the literature [69] determines a priority for each sample in the experience pool, enhancing the likelihood of sampling valuable training examples at the moment of sampling. The error term of the temporal differencing algorithm is used to construct the sample priority. The greater the absolute priority value, the greater the sample's probability at the time of sampling. The results of the experiments show that this approach has a faster training speed and better runtime results.

A DQN based on a competitive architecture is proposed in [70], with the critical change replacing the fully connected layer after the convolutional layer of the CNN with two branches, one of which fits the state value function and the other the action dominance function. Finally, the Q function values are formed by adding the output values of the two branches. Experiments demonstrate that this enhancement can more precisely predict the value function value. CNNs, which are used in DQN, cannot recall for long periods. As a result, the study in [71] suggested a DQN algorithm (DRQN) that incorporates RNNs. This method adds a recurrent layer (LSTM unit) after the CNN's convolutional layer, allowing it to remember past data. In the fully cooperative algorithm, each agent's reward function is the same, indicating that all agents work hard to attain a common goal. Team Q-learning [77] and distributed Q-learning (Distributed Q -learning) [79] are examples of representative algorithms. The agent's reward function in a fully competitive algorithm is the opposite. There are usually two downright opposed agents in the environment. The agent's goal is to maximize its rewards while minimizing the rewards of the other party as much as possible. Its representative algorithm is Minimax- Q [80]. In mixed tasks, there is no deterministic positive or negative relationship between the agent's reward function. This model is appropriate for selfish agents. In general, the concept of equilibrium solution in game theory is important to solving such tasks. When a state in the environment has various equilibriums, the agent must unanimously choose the same equilibria. This approach is primarily used for static tasks. Nash Q-learning [81], Correlated Q-learning [82], Friend or Foe Q-learning [82][83] are some instances. The algorithm we use in this thesis is fully cooperative.

In this thesis, we propose a DERs discharging scheduling model with centralized learning and decentralized execution. Most early MARL algorithms used one of two training modes: centralized or decentralized. The centralized training employs a single training network to oversee the entire learning process, making it easy to overfit and computationally intensive. The decentralized training uses multiple training networks, each completely independent of the others, making the algorithm challenging to converge due to the lack of a centralized objective function. As a result, both training modalities are limited to small systems with a few agents. Centralized Learning and Decentralized Execution (CLDE) [84] combines the characteristics of the above two modes. On the one hand, agents obtain global information based on mutual communication for centralized training and then perform decentralized execution based on their partial observations. The most significant benefit of CLDE is that it allows additional information (such as the global state of the environment, actions, or rewards) to be added during training that would otherwise be ignored during the execution phase. It allows for real-time control and guidance of the agent's learning process.

2.3 RL in Distributed Energy Resource Management

Several existing approaches studied the application of reinforcement learning to help in the decision-making process while using multiple DER. The work in [16] proposed an RL approach to learning EV charging behaviour, which is determined by a predefined heuristic scheme. They use collective EV fleet control actions rather than individual EV control operations. Defining an MDP for the entire EV fleet minimizes the state and action space. A simple heuristic is employed to translate collective control activities back to individual control actions. A cost-effective day-ahead plan is learned based on historical data of cooperative control operations, which automatically considers the heuristic division strategy. We also utilize fitted Q iteration instead of temporal difference learning to deal with continuous variables in our state and action space. O'Neill et al. [31] developed an RL-based Consumer Automated Energy Management System (CAES) for adjusting an individual home consumer's energy use to reduce residential energy costs and flatten the daily energy consumption curve. Both energy pricing and customer decisions are treated as MDPs in their approach.

Compared to the uncontrolled instance, this strategy was able to cut a consumer's costs by 16-40% in simulations. CASES only proposed the system for one consumer; therefore, state and action spaces are constrained. The authors of [32] present how to model resources such as battery energy storage systems, solar generation systems, directly controllable loads, load shedding, programmed deliberate islanding, and power curtailment mathematically in the microgrid optimal scheduling problem. The suggested modelling also includes a methodology for determining the availability cost of battery and solar system assets. They model the Battery Energy Storage System (BESS) state of charge as a non-recursive constraint to the multi-integer linear programming issue. The paper presented a practical approach to computing the BESS availability cost in USD/kWh modelling of the BESS system considering battery, converter, and transformer relevant values of efficiency. The BESS approach does a great job of providing a mathematical approach but might fail with more complex data, which can be addressed using advanced methods like deep learning. The work in [33] presents a multi-agent-based model of distributed energy and the load management approach for DER. In a multiagent system, electricity suppliers and consumers are depicted as autonomous agents capable of making local decisions to optimize their profit. Whether on the supply or demand side, each agent is designed to optimize its utility in an auction-based market using reinforcement learning, allowing it to adapt its behaviour to other agents in a competitive and stochastic energy market. The reinforcement learning capability, which is based on the model-free Q-learning algorithm, enables agents to identify the best policy to maximize their utility without communicating directly with other system entities.

The study in [34] presents a RL-based framework for home energy management (HEM) to achieve effective home-based DR. The authors used a finite Markov decision process (FMDP) with discrete time steps to tackle the hour-ahead energy consumption scheduling problem. They devised a data-driven strategy based on neural networks (NN) and a Q-learning algorithm for the HEM system that provides satisfactory performance on cost-effective schedules. They also provided scheduling decisions for household appliances and electric cars (EVs) by a proposed framework, which has a dual goal of minimizing the power bill and the DR-induced discontent.

Since data privacy is really important in this age, there has also been research to protect users' data and secure the system. The authors in [35] propose a Deep Robust Adversarial Reinforcement Learning for privacy process among the users. In the proposed work, the entire model is done in three different steps. The first step develops strategy patterns for the users considering the integration of renewable energy and effective demand response analysis. The second step in the process exhibits the learning process of the consumers using Robust Adversarial Reinforcement Learning for privacy process among the users. The third step develops an optimal strategy plan for the users to maintain privacy among the users. Another interesting approach was discussed by [36], where the authors proposed a decentralized control of residential energy storage system for peak shaving. The model treats the problem as a non-cooperative aggregative game, where the allowable peak acts as the global constraint which has to be satisfied by the community and proposes a decentralized control algorithm to achieve the equilibrium strategy of the aggregative game.

Parallel to the above-mentioned work, several studies such as those in [37] [38] [39] [40], and [41], to name a few, propose models to reduce the peak load. Although these models do not use MARL, they represent the body of work on peak load reduction using methods like direct load control, providing incentives to consumers, optimization techniques and implementing game theory.

Our approach is different from existing studies as it focuses on the peak shaving efficiency of scheduling discharging DER in the local grid, considering the daily forecasting of other consumptions in a local grid. We proposed a centralized training and distributed execution MARL model to provide day-ahead optimized discharging scheduling of DER. The model offers a corresponding scheduling strategy based on the confident interval of daily forecasting learned during the simulated training process. The agents have a cooperative relationship with each other, which allows them to provide coordinated scheduling to maximize daily peak shaving performance in the distributed execution without communication after training. Local agents' distributed execution enhances network connection robustness and improves general DER owners' data privacy from presenting their daily discharging plan collected in a centralized server. We use an actor-critic approach [18] that has two neural networking instead of Q-learning. This advanced technology improves our model's complexity, increasing the learning performance and providing a potential for future distributed training.

2.4 Overall Proposed System Model Architecture

The aim of this subsection is to provide the readers with an overall structure of the proposed system. Specifically, the relation and interaction of the two major components, the prediction system for DR in Smart Grids and the MARL system for the integration of the DERs into the smart grid.

Figure (2.1) shows the system architecture. Each DER (e.g., an EV battery) is modelled as an agent in our system. The agent has the following properties: Capacity, Discharging rate, Plug-in Time. The capacity is the total energy allowed to discharge back to the grid of the DER's battery. The discharging rate is in W during the discharging period. The

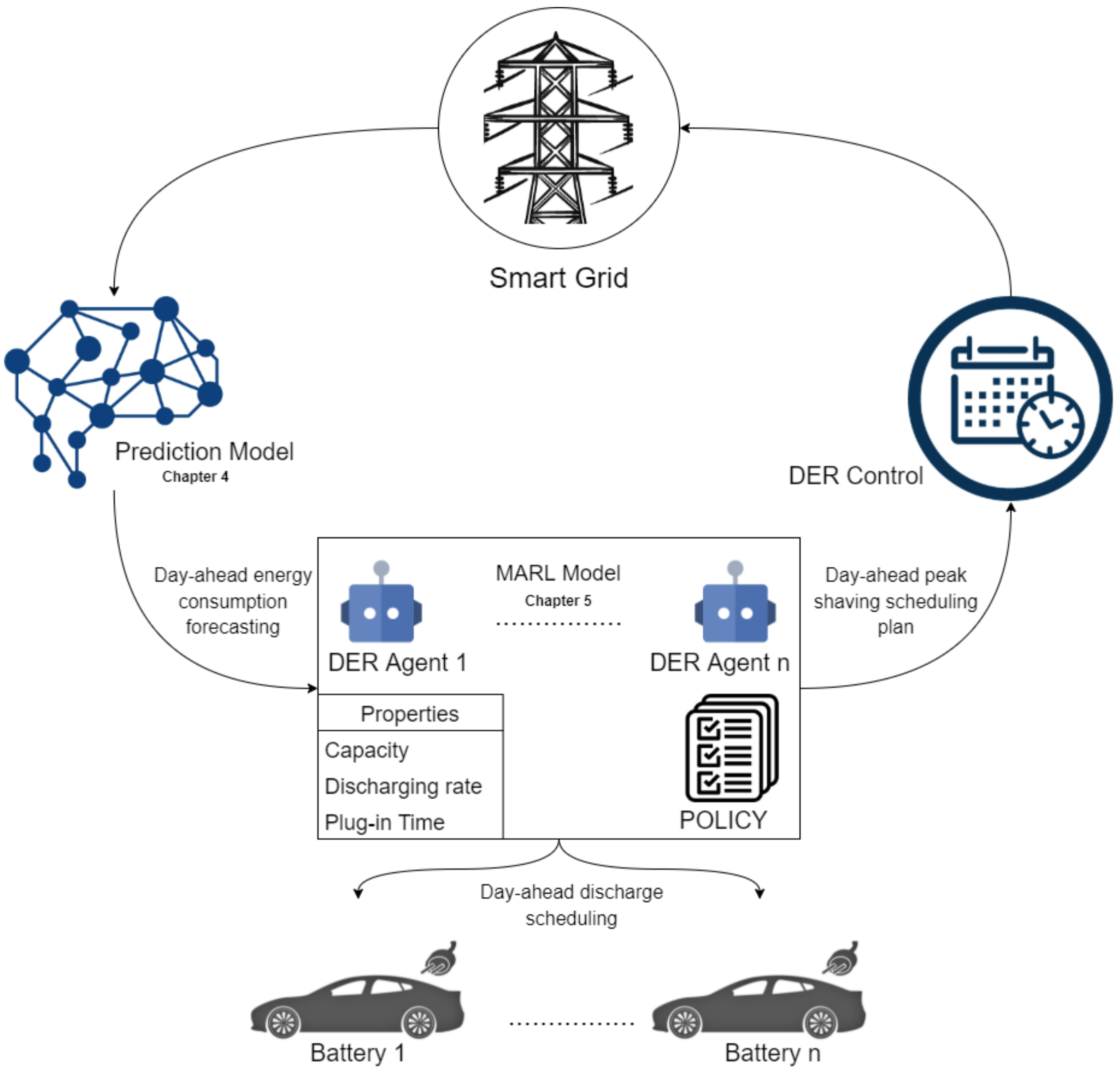


FIGURE 2.1: System Model of Day-ahead Discharging Scheduling

division of capacity by discharging rate defines an agent's expected discharging time per day. The plug-in time represents the time range of the battery attached to a charging station and ready for discharging events. The day-ahead discharging will schedule in the corresponding agent's plug-in time.

The subsections below provide a brief description of the two components.

2.4.1 Prediction system

The prediction system contains the following component:

- An image clustering system with the images of the daily power consumption curve generated by the dataset. This system provides the type of each day based on the image-processing technique. The type of each day affects the selection of the training set.

- An ensemble model for power consumption prediction combines four machine learning algorithms. The machine learning algorithms include Cubist, FNN, Random Forest and LSTM.
- An dynamic weighting mechanism further improves the accuracy of the predicted daily consumption curve. Moreover, a majority voting mechanism improves the predicted peak point time.

2.4.2 MARL

The MARL system takes the output of the day-ahead daily power consumption from the prediction system and outputs a day-ahead discharging schedule for each DER. The proposed MARL model is trained on a dataset of historical daily prediction and actual power consumption. The RL training process increases the model performance on the day-ahead scheduling and optimizes the daily peak shaving. We use the actor-critic RL model with a centralized training and distributed execution design.

The following chapters provide the details of the design and validation of the prediction and the MARL models.

Chapter 3

Image-based with Peak Load Ensemble Prediction System for Demand Response in Smart Grid

3.1 Introduction

In an energy management system, load forecasting is a crucial component. An intelligent energy management system demands short-term peak load forecasting to engage loads from smart homes and EV battery systems. This is because these battery systems are relatively small; thus, they require shorter charging and discharging time [1][2], and [3]. The energy management system allows general users to enroll in load balancing by charging or discharging the electric vehicle battery during peak and non-peak times based on a proposed time. Thus, an accurate peak load prediction model has a significant impact on developing an appropriate mitigation strategy toward load balancing [4] and resource planning effectiveness [5]. This thesis aims to design a power consumption forecasting system focusing on accuracy during peak periods using a novel image processing technique based on the characteristics of the load curve. To this end, we use a convolution neural network model to extract the features of the image and use the k-means mechanism for clustering. Although the consumption value in kWh of each day is different, the shape of the curve almost exhibits a similar pattern for each day. As a result, we use image processing to separate similar patterns of power consumption curves to different groups. To validate our hypothesis, we use the energy consumption data from Thunder Bay (Ontario, Canada) area and test in 900 days, including 200 shoulder-season days, to show the improvement of peak prediction accuracy.

3.2 System Model

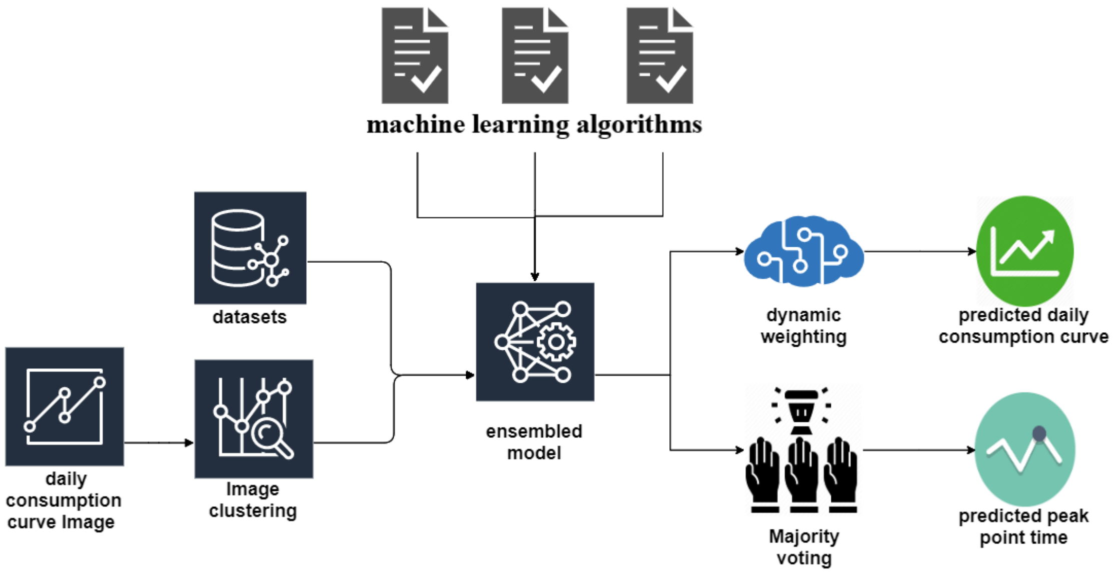

machine learning algorithms
datasets
daily consumption curve Image
Image clustering
ensembled model
dynamic weighting
predicted daily consumption curve
Majority voting
predicted peak point time

FIGURE 3.1: The System Model. Daily consumption curve images, clustered and fed into the ensemble model for prediction

Figure (4.2) shows the proposed system model. The system classifies every training day to a type of pattern based on an image clustering technique. Then, the training set is selected depending on the type as the predicted day and trained using multi algorithms ensemble model. The ensemble model includes 5 prediction algorithms (explained in subsection (3.2.3), Random Forest, Cubist, XGBoost, LSTM, and FNN. The results will be averaged through a dynamic weighting mechanism that outputs a predicted curve and a majority voting mechanism to determine the peak point's output.

3.2.1 Weather Data Processing

All weather information is collected from the Dark Sky API of Thunder Bay, ON, Canada. The 5-min interval weather dataset, as shown in Table (3.1), is the training set of the ensemble prediction model. Table (3.1) records weather and time information and power consumption(output) every 5 minutes as each row. The weather information provided by Dark Sky API was originally recorded hourly; we equally separate the one-hour weather parameters into the 5-min interval. The first two rows record the historical data consumption one day and two days ago. The power consumption data are collected in about 150 houses of the residential areas in Thunder Bay provided by Synergy North, including the market and small businesses. In the variable time, we convert the data-hour-minute structure to a Sin and a Cos variable to represent hours and minutes, respectively. Hours and minutes are cyclic, so Sin and Cos make the cyclical interpretation of time easier. Compared to the date-hour-minute structure, this new data format allows us to extract more information because it becomes easier to identify the time of the day; for example, 23:00 is very close to 0:00, but in numerical expression, they are 82800 seconds and 0 seconds. After we change time to cos & sin expression, time data act as a clock; 23:00 and 0:00 are numerically close to each other. The daily dataset, as shown in Table (3.2), includes weather and time information for every day. The temperature difference represents the average temperature change compared to the previous day. It is a factor correlated to detecting season-changing. After image clustering power consumption curves in the training set, the daily dataset is applied to classify which group the predicted day belongs.

Table 3.1: 5-min interval weather information dataset

Attribute No.	Description
1	power consumption 1days ago
2	power consumption 2days ago
3	Sine expression of hours
4	Sine expression of minutes
5	Tempreture(Celsius)
6	Dew
7	Humidity
8	Wind speed
9	Visibility
10	Pressure
11-17	Day of week(dummy variable)
18	Power Consumption(output)

Table 3.2: Daily weather information dataset

Attribute No.	Description
1	Average temperature
2	Temperature difference
3	Wind speed
4	Dew
5	Humidity
6	Pressure
7	Visibility
8	Sunrise(Time in seconds)
9	Sunset(Time in seconds)
10-16	Day of week(dummy variable)
17-28	Month(dummy variable)
29	Group of day(Output: 0,1,2)

3.2.2 Phase 1: Classification

In this subsection, we use image processing techniques to classify the images of daily consumption curves into different types and select the previous days in the same type of predicted days as the training set. The daily power consumption curve images should be a plot with the x-axis in 5-min intervals and the y-axis as power consumption in kWh. The graph's format is as follows: images dimension is 224×224 pixels (RGB); bold curve line to emphasize information; the x and y-axis labels are deleted. Example images as shown in Figure (3.4),(3.5), and (3.6).

Visual Geometry Group or also called OxfordNet (VGG16) CNN structure, which pre-trained with ImageNet (open source image dataset), is powerful to extract features from images [53]. Transfer learning from the pre-trained model can solve the shortage of daily power consumption curve images. The idea of a pre-trained model is to accumulate knowledge in a model trained for a specific task and transfer it to other relevant tasks. The ImageNet dataset is an object detection that requires the model to classify an input image into 1,000 different classes. The VGG16 pre-trained data is used to extract the feature of objects, similar to the feature extraction of curve images. The saved weight of filters in the VGG16 are transferred to extract the feature of images of the daily consumption curve.

The VGG16 model, as shown in Figure (3.2), is a CNN with five blocks and a total of 16 layers. Each block has two to three convolutional layers to extract features and one max pooling layer to reduce half of the size. The convolutional layer has several random filters to extract different features and enhance themselves during the training process. In VGG16, three 3×3 convolution kernels are used instead of 7×7 convolution kernels, and two 3×3 convolution kernels are used instead of 5×5 convolution kernels. By doing so, under the same perceptual field, the network becomes more in-depth; thus, the effect of the neural network improves. The original image data format has a large dimension of numbers, which is too complex to apply k-means. VGG16 model extracts image features and converts each image to an array of numerical numbers. In the first two layers in VGG16, kernel size is $224 \times 224 \times 64$, which means there are 64 feature maps in this layer, and each feature map represents a 224×224 image after convolution. Each feature map is calculated by the convolution of previous layer data and a filter. (in VGG16, all convolution filters are 3×3) For the image I convolute with filter k, the formula as following:(k represent the size of the filters, for size of filter as 3, the $\sum_{v=-k}^{k}$ will be $[-1, 0, -1]$)

$$(I * K)_{ij} = \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i-u, j-v)K(u,v) \tag{3.1}$$

There are three fully connected layers in the pre-train model in ImageNet. The first two layers have 4096 channels output of the vector representation of an image; the third contains 1000 channels(one for each class), which is a softmax layer output of the image class. It is unnecessary to keep the last softmax layers since the output of curve image classification is not 1000 class. Due to the limit of a relatively smaller dataset, it is better to keep the pre-trained weights of filters. If a larger dataset is available, the pre-trained model can be fine-tuned by reconstructing fully connected layers and unfreezing some convolution layers to retrain the model. By doing so, the model will be more fitted to the task. The reason for using CNN to extract features from daily power consumption images instead of using RNN to extract features directly from numbers: we want to cluster our datasets based on the pattern of the curve or not the digital differences at each time of day. For example, when using RNN to extract features directly from time-series data, one day with a smooth curve with a peak at 5 pm may have a closer distance to a sharp peak at 5 pm than a sharp peak at 8 pm. In our hypothesis, we want to cluster the days with sharp peaks together because we believe these days are more correlated in the training process.

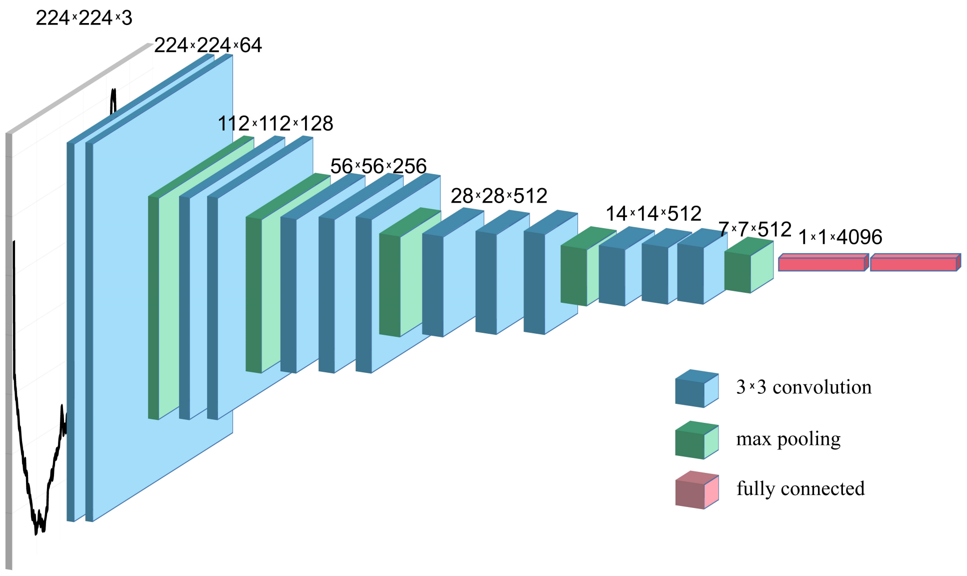


FIGURE 3.2: The structure of VGG-16 convolutional neuron network.

Classification of daily power consumption curve is an unsupervised image clustering; thus, the number of clusters needs to be selected wisely. The k-means algorithm calculates the distance between arrays and separates closer arrays into different groups after applying the k-means algorithm with the number of clusters from 2 to 10. The average distance between each image and its clustered center is shown in Figure (3.3). The average distance dramatically decreased from 2 clusters to 3 clusters, and after 3 clusters, the average distance decreased smoothly. It means that separating the images into three

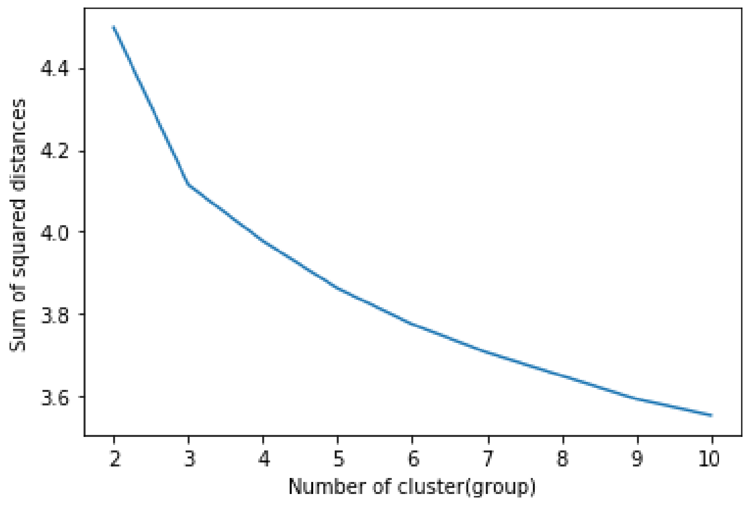


FIGURE 3.3: Relationship between number of cluster (group) and sum of squared distances of samples to their closest cluster center

types is most appropriate for our datasets. A relatively smaller number of clusters can help the accuracy of forecasting the cluster of new predicted days and increase model robustness toward new unseen data. The optimal selection of the number of clusters might vary for different datasets. The seasons in Thunder Bay are mostly separated into three; spring/summer is short (4 months), which the people often consider one season. This may be the reason why the number of clusters is 3. A larger number of clusters may cause less difference between each cluster; a smaller number of clusters may cause more feature differences inside each cluster. It is important to choose an appropriate number of clusters for the datasets.

After image clustering of 900 days, 45% percent of the daily consumption curve belong to type 1, 25% belong to type 2, and 30% belong to type 3. Examples of each type as shown in Figure (3.4),(3.5), and (3.6), images of type 2 have a sharp peak in the afternoon, in type 1 the curve oscillations occur at noon and in type 3 the curve has a long flat peak. Type 1 and 3 do not have a sharp peak; thus, predicting the exact peak time is difficult. Before predicting the power consumption of a new day, the system should first classify the type of curve of new days. This is a typical supervised machine learning problem in which we could use the historical clustered days with parameters in Table (3.2) as a training set. The type of new day is predicted by the random forest model, as known as the ideal solution of the classification task, with 400 ntrees and set the max depth as 9 using parameters in Table (3.2). The accuracy of leave-one-out cross-validation in 900 test cases achieve at 98.36%. Leave-one-out cross-validation means, in each testing case, only use the data before the testing day as the training set. The same validation is used in testing the performance of the prediction model. We hypothesize that the power consumption primarily correlated to the previous days in the same type of curve

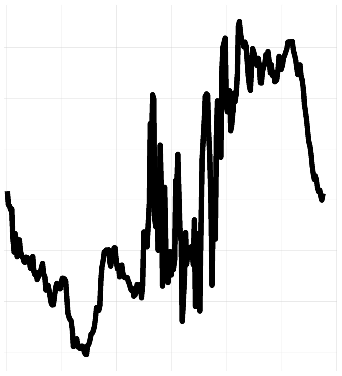


FIGURE 3.4: Type 1: a smooth peak in afternoon with a random oscillations at noon.

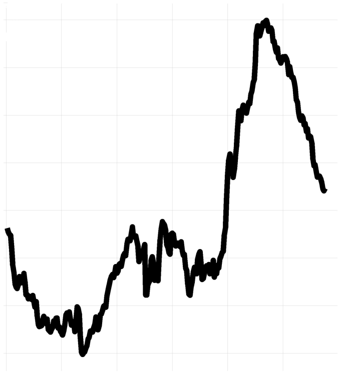


FIGURE 3.5: Type 2: a lower consumption in morning and a sharp peak in afternoon.

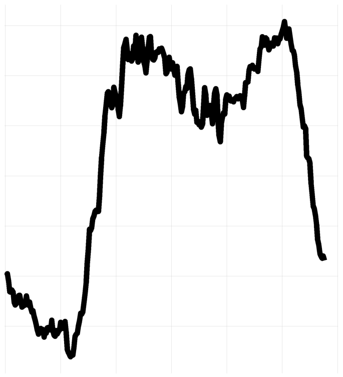


FIGURE 3.6: Type 3: a long flat peak start at morning and end afternoon, may have a sharper peak in afternoon.

as the predicted day. The selection of a training set of power consumption forecasting models should contain all the days with the same curve type as the predicted day in the last 60 days. During testing, using data further than 60 days ago as the training set will decrease the model's performance; this number might differ in different geographical areas.

3.2.3 Phase 2: Ensemble Prediction Model

Ensemble learning integrates a series of classifiers with general performance into a combined classifier with better performance. Because training a single classifier with good performance may be difficult. The general structure of Ensemble learning is: first generate a group of "individual learners" and then use a particular strategy to combine them. Individual learners are usually generated from training data by an existing learning algorithm, such as the decision tree. If the integration contains only the same type of individual learners, for example, "decision tree integration" contains all decision trees, such integration is "homogeneous." The ensemble can also include different types of individual learners, such as a decision tree and a neural network at the same time. Such an ensemble is "heterogeneous." Ensemble learning is sometimes named differently, and sometimes it is called multi-classifier system, committee learning, modular systems, classifier fusion, combination, aggregation. These concepts are related to each other, but there are some differences. The industry has not yet reached a consensus on the definition of the concept. The performance of the whole algorithm is compelling, and it is the first choice in many high-level competitions (Knowledge Discovery and Data Mining, Kaggle). Different learners have different preference models, but each is a weakly supervised model. Ensemble learning combines multiple weakly supervised models to obtain an outstanding strong-supervised model. The idea is that different learner's correct errors mutually to achieve the ultimate accuracy improvement.

The combination of learners may benefit from three aspects: First, from a statistical point of view, because the hypothesis space of learning tasks is often ample, there may be multiple hypotheses that achieve the same performance on the training set. If using a single learner, it may have poor generalization performance due to misselection. Combining multiple learners will reduce this risk. Second, the single learning algorithm falls into a local minimum and some local extremes from the calculation perspective. The generalization performance corresponding to the local minimum may be terrible, and the combination of multiple trained models can reduce the risk of falling into a bad local minimum. Third, from the representation perspective, the hypothesis of some learning tasks maybe not be in the hypothesis space considered by the current learning algorithm. Combining multiple learners makes it possible to learn a better approximation due to the expansion of the corresponding hypothesis space.

There are two conditions to determine if the performance of the ensemble learner is better than that of the individual learner: the individual learner is better than the result of random guessing, and individual learners must be independent of each other. The first condition is relatively easy to implement. Train a model in the present which produce results that are usually better than guessing. The second condition is the core issue of integrated learning research. Each learner learns the same problem, so individual learners cannot be completely independent of each other. The entire algorithm learning process is from data to model to output. There are three approaches to enhancing the diversity of individual learners. First approach considers the input data. Each learner learns from different samples will produce a diversity of relatively different individual learners. Each learner selects samples randomly or uses different attribute subsets to train different individual learners are the most common way to divide the training samples. The second consideration is the model. Suppose the model of the base learner is different, then they will train different individual learners from the same training set. The third approach is the output. If we divide according to the characteristics of the label, we can also get different individual learners. In this thesis, we use five machine learning models: Random Forest, XGBoost, Cubist, Feedforward Neural Network, and Long-Short Term Memory.

3.2.3.1 Random Forest

In the 1980s, Breiman et al. invented the classification tree algorithm [85]. By repeatedly dichotomizing data for classification or regression, the tree algorithm significantly reduces the amount of calculation. In 2001, Breiman combined the classification trees into a random forest, which is to randomize the use of variables (columns) and the use of data (rows), generate many classification trees, and then aggregate the results of the classification trees. Random forest improves the prediction accuracy on the premise that it does not significantly increase the amount of calculation. Random forest is not sensitive to multiple factor linear regression problems, and the results are relatively robust to missing and unbalanced data. It can predict the effects of up to thousands of explanatory variables well and is currently one of the best machine learning algorithms.

Random forest, as the name implies, is to build a forest randomly. There are many decision trees in the forest, and there is no correlation between each decision tree in the random forest. When a new input sample enters, let each trained decision tree in the forest judge separately to see which category the sample belongs to (for the classification algorithm). The prediction of that sample belongs to the category selected the most. The random forest can handle both discrete-valued quantities and continuous-valued quantities. In addition, the random forest can also apply to unsupervised learning clustering and outlier detection.

The decision tree is a tree structure (can be a binary tree or non-binary tree). Each non-leaf node represents a test on a characteristic attribute. Each branch represents the output of this characteristic attribute in a specific value range. Each leaf node stores a category. The process of using a decision tree to make a decision is to start from the root node, test the corresponding feature attributes in the items to be classified, and select the output branch according to its value until the leaf node is reached. The category stored in the leaf node is the decision result. Decision trees are based on the calculation of entropy (3.2) and information gain (3.3) of each variable p.

$$Entropy\ H(X) = -\sum p(X) \log p(X) \tag{3.2}$$

$$Information\ Gain\ I(X,Y) = H(X) - H(X|Y) \tag{3.3}$$

Random forests have many advantages: it performs well on data sets, and introducing two randomness makes random forests less likely to fall into overfitting. It has excellent anti-noise ability. It can handle very high-dimensional (many features) data without select features and has strong adaptability to data sets: it can handle discrete and continuous data. The data set does not need to be standardized. The training speed is fast, and the importance of variables can be ranked based on data correlations. During the training process, it can detect the mutual influence between features. The implementation is relatively simple. The disadvantages of the random forest are that it is easy to fall into overfitting in some sample sets with relatively large noise. Features with more value divisions are likely to have a more significant impact on the decision-making of the random forest, thereby affecting the effect of the fitting model.

3.2.3.2 XGBoost

The full name of XGBoost is eXtreme Gradient Boosting, which is an optimized distributed gradient boosting library designed to be efficient, flexible and portable [86]. XGBoost is a tool for massively parallel boosting trees. It is currently the fastest and best open-source boosting tree toolkit, ten times faster than common toolkits. In terms of data science, many Kaggle members choose XGBoost for data mining competitions, which are commonly found in major data science competitions. In terms of large-scale data in the industrial world, the distributed version of XGBoost has extensive portability. It supports Kubernetes and Hadoop, SGE, MPI, Dask and other distributed environments to solve the problem of large-scale data in the industry.

XGBoost algorithm is based on the decision tree and uses Gradient Boost as a framework. XGBoost also uses an additive model and a forward step-by-step algorithm to realize the optimization process of learning, but it is different from gradient boosted decision trees (GBDT). The main differences include the following: In the objective function, the loss function of XGBoost adds a regularization term to control the complexity of the model. The regularization term contains the number of leaf nodes of the tree, the weight of each leaf node and the sum of squares of the score values of the points. GBDT only uses first-order derivative information during optimization, and XGBoost uses first and second-order derivative information. XBGoost handles missing values and automatically selects the optimal default segmentation direction for missing values through the learning model. In addition to adding regular terms to prevent overfitting, XGBoost also supports row and column sampling to prevent overfitting. It can get better results with fewer computing resources in the shortest time.

The XGBoost algorithm is an implementation of gradient boosted decision trees designed. This algorithm has a high performance with structured or tabular datasets on classification and regression predictive modelling problems. Energy forecasting is a complex problem; we should approach it from both the classification and regression sides.

3.2.3.3 Cubist

The Cubist model tree algorithm is a binary regression tree model where the last nodes are the linear regression functions that can produce continuous numerical attributes [87]. The M5 algorithm uses a divergence metric to produce a decision tree. The next step to developing a tree model involves tree pruning, tree evacuation and substitution of trees with linear regression functions. In the end, this method produces a tree-like structure with the linear regression model. The advantage of this algorithm is that the model trees are generally much smaller than regression trees with high accuracy.

3.2.3.4 Feedforward Neural Network

Feedforward means that all information is finally output to y through some intermediate calculations from the input, and there is no feedback from the model's output to the input [88]. For the case of feedback, it is a recurrent neural network. The feedforward neuron network has been widely used in the industry. It is also the basis for understanding the widely used recurrent neural network in natural language processing.

The network represents that the model is a model formed by combining different basic functions. For example, the final function $f(x) = f^{(1)}(f^{(2)}(f^{(3)}(x))), f^{(1)}$ is the first layer of the network, $f^{(2)}$ is the second layer of the network, and so on. The length of this chain is also called the depth of the network, hence the name of deep learning. The last layer of the feedforward network is called the output layer. For the training data, each input x has a corresponding label $yf^{*}(x)$, and the output layer of the network. The result should be as close to y as possible. Nevertheless, for other layers, there is no such direct correspondence with the training data. The algorithm only requires the final output to be close to the actual mark, and the purpose of each layer in the middle is not clearly defined, so these layers are also called hidden layers. The hidden layers also make the neuron network a black-box algorithm; hence it is hardly explainable.

Feedforward Neural Network(FNN) is a simple implemented artificial neural network without any cycles or loops. In this network, the data only move one-directional from the input layer to the output layer and passes many hidden layers in between.

3.2.3.5 Long-Short Term Memory

Long short-term memory (LSTM) is a special RNN, mainly to solve the problem of gradient disappearance and gradient explosion during long sequence training [89]. In general, LSTM can perform better in longer sequences than ordinary RNNs. Besides the information moving forward, the recurrent neural network also learns from the predicted output and the actual output. After each forward pass through a network, backpropagation performs a backward pass while adjusting the parameters of the model (weights and biases). During backpropagation, each weight is updated through gradient descent. The network learns through constant forward pass and backpropagation. The forgot gate in LSTM helps abandon details from the block so that only essential information is retained in the cell to make the computation efficient and effective.

There are three main stages inside LSTM. First is the forget stage, which is mainly to forget the input passed in by the previous node selectively. Generally speaking, forget the unimportant information and remember the important ones. The second is the memory stage. This stage selectively remembers the input of this stage. Mainly to select and memorize the input. What is essential is recorded, and what is not essential is recorded less. The third stage is the output stage which will decide which outputs will be considered as the current state.Because of the increasing contents, the number of parameters has increased, and the training difficulty has also increased a lot.

3.2.4 Phase 3: Dynamic Weight Calculation

RMSE is one of the most convenient error calculations that could not accurately represent peak prediction performance. This thesis focuses on the correctness of the peak period prediction; RMSE also includes the morning's error. The morning power consumption is majorly random and impossible to predict, leading to a massive increase in RMSE. The peak point time difference between predictions and actual could only be a reference for judging a model's accuracy; it can not represent the error either. On some days, there may be two peak points more than one hour away from each other with similar power consumption (kW); one of them will be the exact peak; a model could predict a curve that fits two peaks but captures the other as the exact peak. Even though the peak prediction point is far from the actual, the model performs with excellent accuracy. Thus, we use a relative root mean square error (rRMSE). In equation (3.4), RMSE is divided by the average power consumption of the tested day, between 1:00 pm and 12:00 pm. The rRMSE represents the ability of a model to predict the peak period. In the following formula, n represents the number of energy consumption in the array, y_i and x_i are the predicted and actual power consumption receptively.

$$rRMSE = \frac{1\sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}}{\frac{1}{n}\sum_{i=1}^{n} x_i} \tag{3.4}$$

The ensemble model aggregates three model algorithms' predictions and produces a final prediction for the unseen data. Instead of using direct averaging, we can use a weight calculation mechanism to assign a weight to algorithms based on their accuracy. Because the prediction is one day before the predicted day, we can calculate the weight based on the day's peak period's model accuracy before the predicted day. In equation (3.5), M_n represents rRMSE of the peak period between the prediction model results and the real peak time of one day before the predicted day of each algorithm. This mechanism's basic idea is to enlarge the weights of the model, which provides a more fitted peak period curve previously. We use the reciprocal of the difference of time prediction because the smaller error a model has, the higher the accuracy produced.

$$W_i = \frac{\frac{1}{1+M_i}}{\sum_i^n \frac{1}{1+M_n}} \tag{3.5}$$

The improvement of applying weight-based averaging in terms will be presented in the section "Analysis of Results". The system also includes a majority voting technique to output the exact peak point prediction as a reference. The mechanism of majority voting, as shown in Algorithm (1). First, calculate the mean of predicted exact peak points of each algorithm. If there is any predicted peak point that is more than 60 min away from the calculated mean, remove the one with the largest absolute difference and recalculate the mean. It can prevent any model with a worse prediction from ruining the whole system. As mentioned above, there are someday has two peaks; it is doubtless that one model prediction will occur around the lower peak. In this case, three algorithms voting to one peak point will provide much better results than a simple averaging prediction, which is most likely in the middle of two peak points. Dynamic weighting is enough for plotting the curve and scheduling discharging based on the curve. The mean peak point only outputs the exact peak point's times; it can only be considered as a reference.

Algorithm 1 Majority voting of predicted peak point time.

```
    Input: The predicted time of peak point occur in different models: $M_1$,...$M_n$ (type POSIXlt)
    Output: The time in POSIXlt of predict peak point $X$
1: if Any $|M_i - X| >= 60$ mins then
       $X = \frac{\{\sum_{i=1}^{n} M_i\} - \underset{M_i}{\text{argmax}}|M_i - X|}{n-1}$
2: end if
```

3.3 Experimental setup

The experiments are set up based on a historical database provided by Synergy North on the area in Thunder Bay, ON. The weather data provided by Dark Sky API details data processing as described in the above section. The experiments are using leave-one-out cross-validation, in which, on each testing day, we select the previous 60 days as the training set to train our models and test among all days in the dataset. The number 60 is selected based on our initial analysis of the correlation between each basic machine learning model and previous days in the training set. We analyze the average accuracy of all testing cases in the dataset, as shown in the next section of the model performance. The initial approach is to select 60 days to examine the different training day lengths and find the higher correlated period. The number 60 varies for different power consumption datasets based on the local grid sizes, the weather, the development of new electric devices, and the composition of customers, i.e., factories, houses, commercial buildings, etc. The power consumption dataset is recorded in 5-min intervals, which means there are 288 numerical data points in each day. The models are trained with the previous 60 days and output the next-day power consumption contain 288 numbers. This task is a sequence to sequence prediction. All the training sets must be selected before the testing day; otherwise, the model will overfit the future data. To test the effect of selecting days of similar curve structures that are clustered by image-based processing, we select the training set with all the days combined with 50% of days in different types. Other days in the training set are extreme cases; this usually occurs at the beginning of the shoulder season. The number 50% is selected based on an experiment of model performance 25%, 50%, 75%. The higher the percentage we put other type days into the training set, the more negligible effect on the image-based clustering technique. The lower percentage will increase the overfitting of the extreme case. As a result, 50% is a balanced number in our dataset.

3.4 Analysis of Results

The training time of the ensemble model for one-day prediction costs around 5-6 minutes. The time is recorded on the PC with Intel(R) Core(TM) i7-6700 CPU 3.40 GHz, 16GB RAM, and Radeon (TM) RX 480 GPU. In the ensemble model, RF costs around 40-50 seconds, LSTM costs around 3 minutes and 30 seconds, Cubist costs around 30 seconds, FNN costs around 1 minute, and XGBoost costs around 10 seconds. In the running prototype, the prediction model is required to execute at the beginning of each day with fully collected data from yesterday. The execution time of the daily prediction process is the same as the training time since the execution of models takes less than 5 seconds.

The running time of daily image clustering takes 2 hours and 30 minutes for all 900 images in the dataset. The clustering for each day takes around 10 seconds. However, the clustering of the dataset is in the pre-processing phase, which does not infect the running phase. Table (3.3) shows the improvement of the performance of each algorithm by applying image clustering training dataset, dynamic weighting and majority voting during the normal days and season-changing period separately. The season-changing days are manually selected based on the weather report. Each year, there are approximately 14 weeks considered as season-changing period (shoulder-season), also defined as odd days and the rest of the days are so-called normal days. The image clustering refers to the direct averaging of three algorithms after applying our image clustering based training set selection mechanism. Dynamic weighting refers to additional dynamic weight calculation after selecting a training set based on image clustering. The majority voting only regards the peak time. There are three measurements to analyze the performance of each algorithm. rRSME, as shown in equation (3.4), describes the error of fitting the prediction curve to the actual curve. On each testing day, the peak error is the difference between the predicted highest point and the actual highest point. The decreasing number of average peak error increase the effectiveness of discharging event during peak-shaving. Peak time under 60 is how many percentages of days have a peak error under 60 minutes. Any peak error above 60 minutes will go beyond any fully-charged batteries discharging window, which leads to an unsuccessful peak shaving.

TABLE 3.3: Performance analysis on Odd days

	Performance of season-changing period		
	Average rRMSE	Average Peak errors	Peak error under 60
Cubist	0.0947	97.57	34.4%
XGBoost	0.0936	95.62	39.1%
Random Forest	0.0831	82.11	31.2%
FNN	0.1048	86.90	37.67%
LSTM	0.1103	92.07	35.8%
Direct Averaging	0.0858	88.62	32.08%
Image Clustering	0.0819	69.68	46.3%
Dynamic Weighting	0.0764	71.28	43.1%
Majority Voting	NA	71.25	45.3%

During the season-changing period, the final model has a 7.64% rRMSE in the peak period, which reduces 11% error in prediction in the peak period than the model without an image clustered training set. From the perspective of peak shaving, the enhanced model allowed 40% more days successfully enroll in peak shaving activities. The exact peak point prediction can only be a reference for accuracy due to the irregular peak curves. During the season-changing period, the peak may be flat, or we might have multiple peaks. In the flatter peak or multiple peaks, the exact peak point has similar

Table 3.4: Performance analysis on Normal days

	Performance of normal days		
	Average rRMSE	Average Peak errors	Peak error under 60
Cubist	0.0955	39.33	85.4%
XGBoost	0.0935	45.08	79.8%
Random Forest	0.0906	35.32	84.1%
FNN	0.1027	42.31	78.6%
LSTM	0.1186	35.41	83.7%
Direct Averaging	0.0887	32.81	89.7%
Image Clustering	0.0883	31.75	86.3%
Dynamic Weighting	0.0852	32.62	88.0%
Majority Voting	NA	29.55	95.3%

power consumption to a point far from it. This situation leads to errors as the model fails to predict the highest point.

The normal days usually have a sharper peak period with larger power consumption values; thus, the overall number of rRMSE is higher than the odd day. On normal days, the exact peak point is much higher than any other value during the day, so any slight prediction error causes a higher rRMSE value than the odd day (flatter peak). The normal days have less randomness on-peak period, so their peak point prediction performance is more accurate than the season-changing period. During normal days, the season is more certain, and the type of curve remains the same; thus, the image clustering training of the dataset does not have a similar effect as in season-changing periods. We focus on improving the accuracy of hardly predicted odd days; there is only a slight improvement in the accuracy on the normal days and mostly improved by dynamic weighting and majority voting scheme since the prediction of these days is more accessible.

As shown in Figure (3.7), this is a random normal day with the basic ensemble model. Figure (3.8) shows the effect of the applied image-based technique. Figure (3.9) shows the final model in which we apply dynamic weighting and majority voting techniques. A high percentage of days is of normal days; thus, the image-clustering has a minor effect on normal days. The dynamic weighting improves the model accuracy in terms of RMSE.

Odd days are hard to predict for their usage of electricity. As shown in Figure (3.10), the odd day in the basic ensemble model cannot predict the peak accurately and lead to a 1 hour and 45 minutes peak prediction error. Figure (3.11) shows the improvement of applying the image clustering technique. Even so, the model still cannot forecast the increasing usage of electricity; the predicted power consumption curve is more similar to the actual curve. As a result, the peak point is predicted within 1 hour error of the

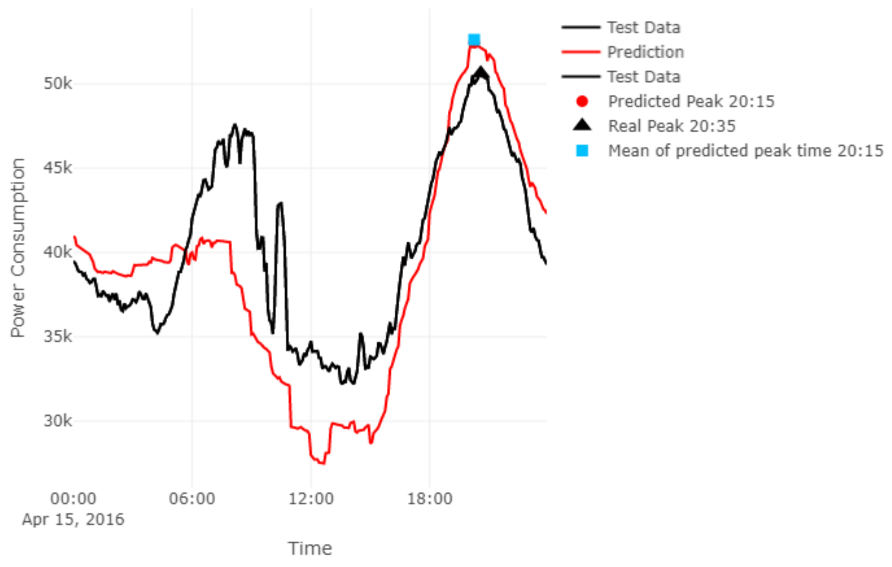


FIGURE 3.7: Basic ensemble model in example normal day

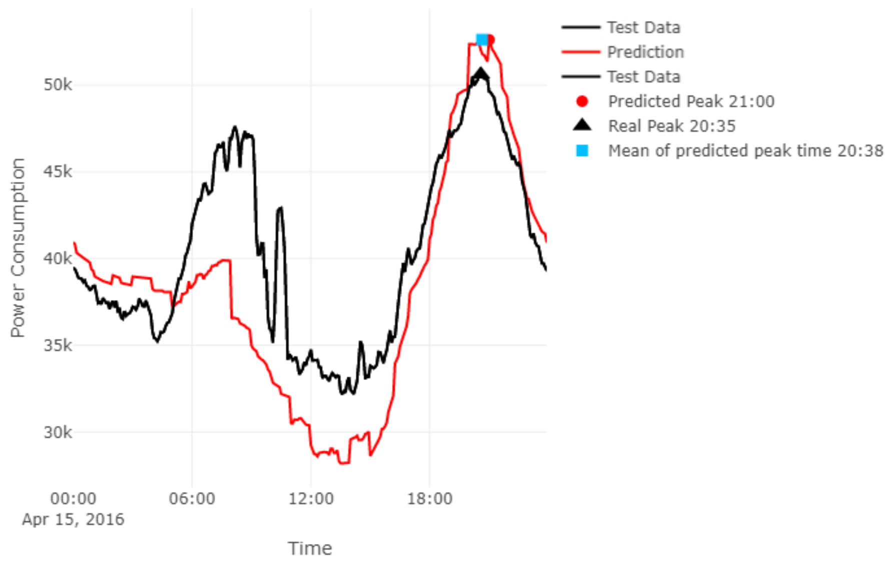


FIGURE 3.8: Basic ensemble model with image clustering technique in example normal day

actual peak time. Figure (3.10) shows the further improvement of dynamic weighting and majority voting.

The Figures (3.13),(3.14), and (3.15) show a detail model performance of random days in each type. The figures present the power consumption prediction of each individual model.

Figure (3.16) shows a detailed model performance of an inaccuracy prediction. The odd days' inaccuracy prediction usually has multiple peaks, and the prediction model only

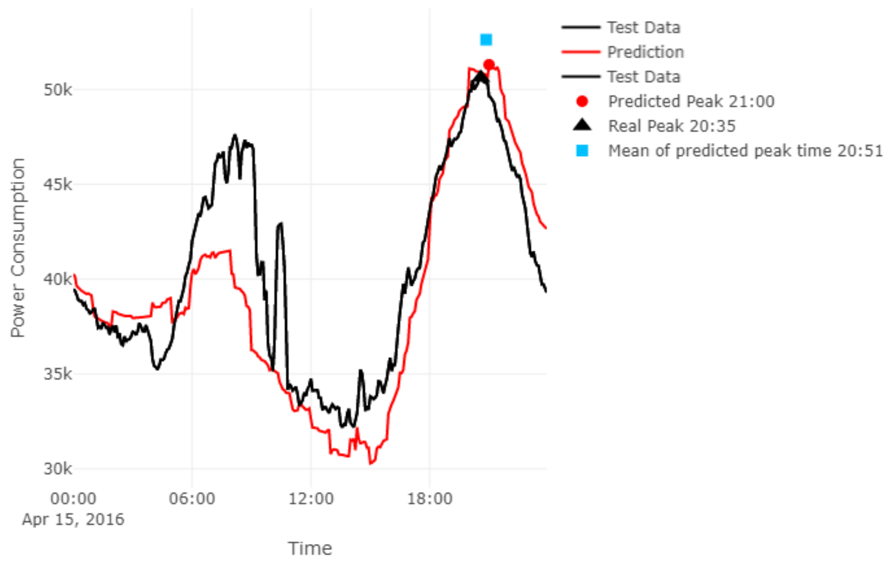


FIGURE 3.9: Ensemble model with dynamic weighting in example normal day

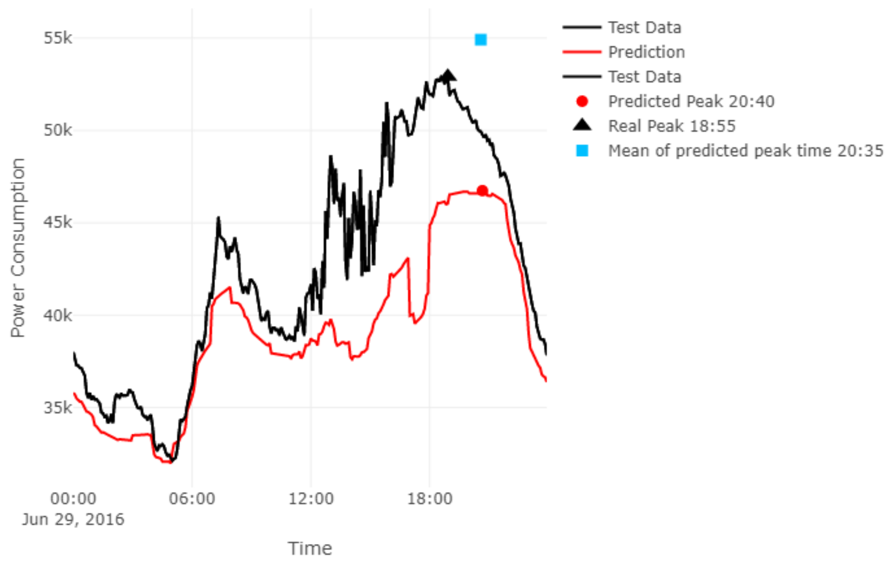


FIGURE 3.10: Basic ensemble model in example odd day

captures one of them. When the predicted peak is not the highest actual peak, it causes a high peak prediction error.

Figure (3.17) show the predictions for one month, including a suggested discharging and charging period. The electricity bills are monthly payments; thus, we analyze the monthly model performance for savings.

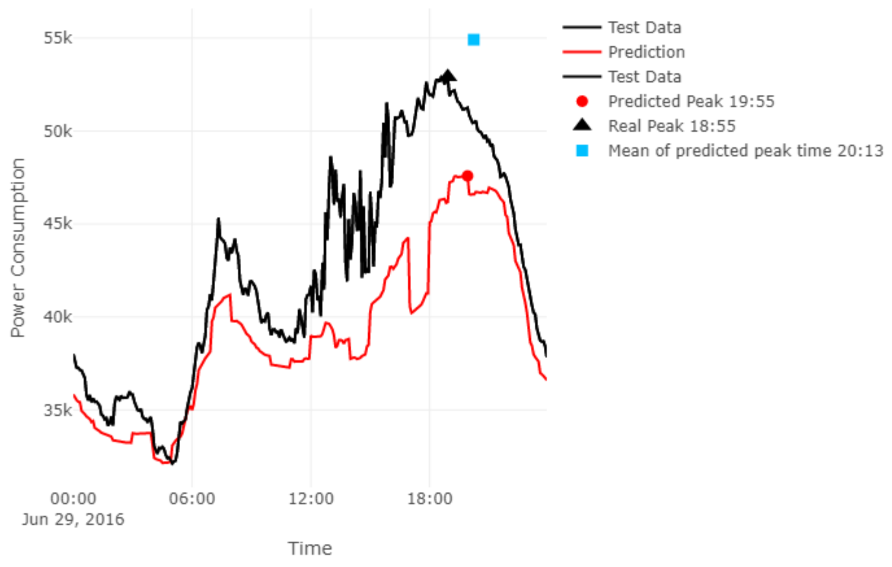


FIGURE 3.11: Basic ensemble model with image clustering technique in example odd day

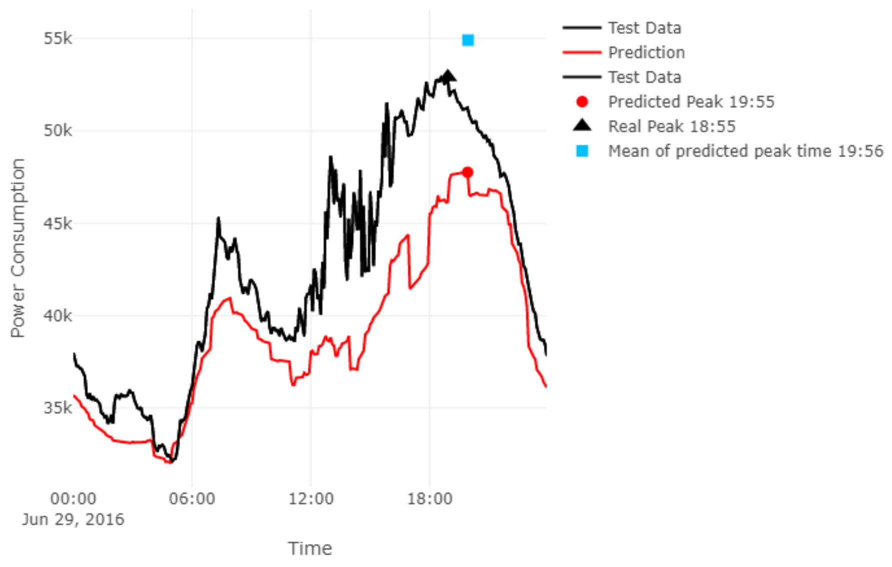


FIGURE 3.12: Ensemble model with dynamic weighting in example odd day

3.5 Discussion

In this section, we present an image processing technique to analyze power consumption curves and make enhanced peak predictions for shoulder season. We proved that the pattern of the daily consumption graph in a smaller residential area could be clustered, and days in similar patterns have a stronger correlation. We also proposed a dynamic weighting and majority voting mechanism to further improve the ensemble system's performance. The system enhances the demand response planning during the shoulder

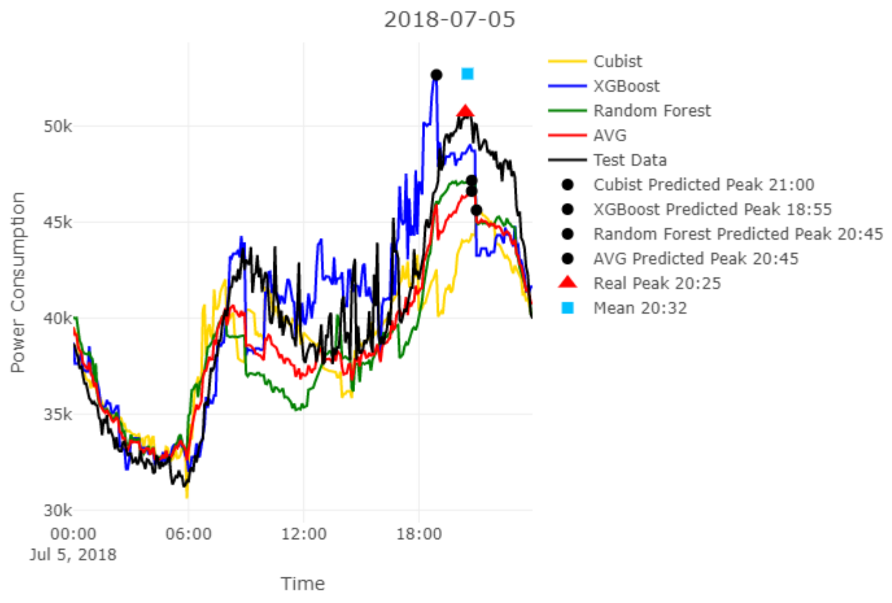


FIGURE 3.13: The detailed model performance of day in type 1

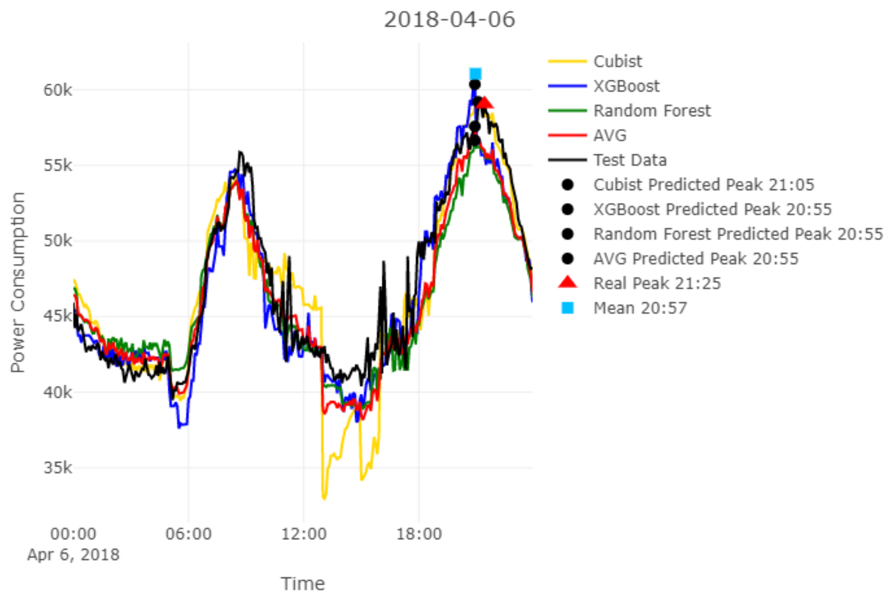


FIGURE 3.14: The detailed model performance of day in type 2

season and increases peak load period prediction accuracy. On the other hand, we still need to exterminate if image-based processing could apply to other locations.

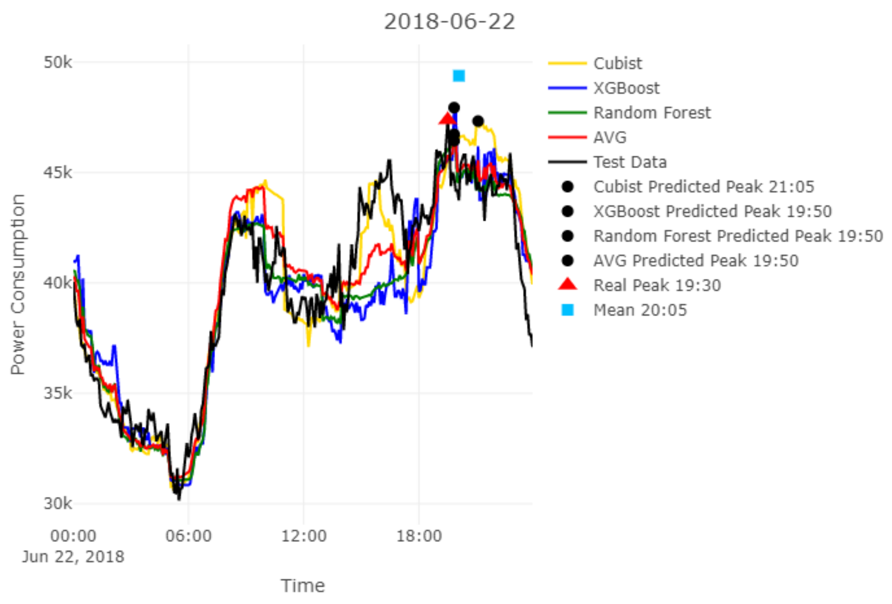


FIGURE 3.15: The detailed model performance of day in type 3

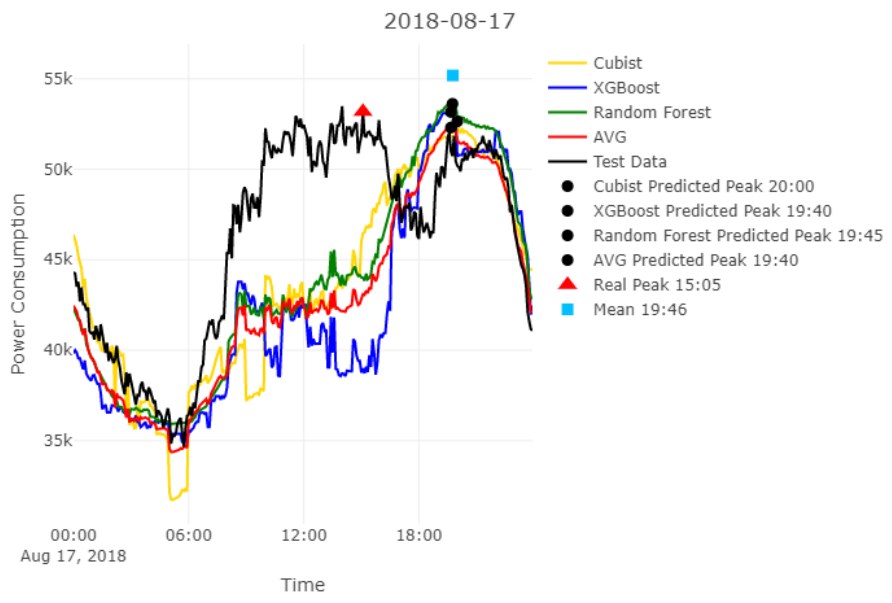


FIGURE 3.16: The detailed model performance of an inaccuracy prediction

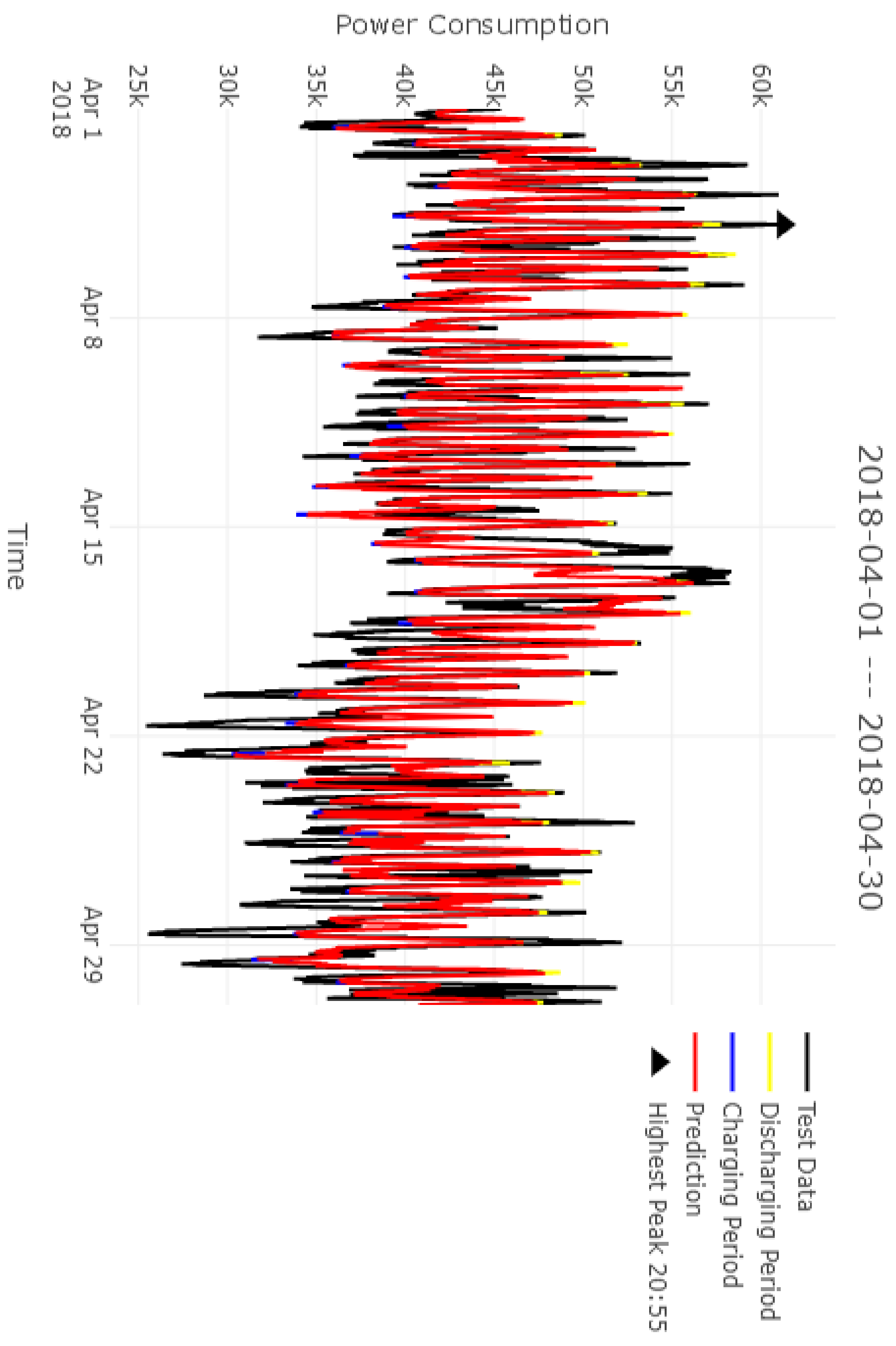


FIGURE 3.17: Monthly report include prediction and suggest discharging,charging time

Chapter 4

Multi-agent Reinforcement Learning for DER Discharging Scheduling

4.1 Introduction

The proposed model in this chapter focuses on the coordination of DERs, especially EVs, to reduce the peak and avoid creating relative peaks. Several researchers have attempted to solve the DER coordination problem using RL. However, most of these studies focus on the uncertainty of the environment [15], the scaling of the system by increasing the number of agents [16], and the comfort of the users [17]. Our work is completely different from previous attempts because we tackle the day-ahead coordination of the DR problem. To this end, scalability is assured since only selected DERs will be engaged based on the previous reward. Our system depends on an external prediction mechanism to determine the day-ahead peak period but provides discharging scheduling of DERs with an optimized peak shaving performance while considering the inaccuracy of energy prediction. The model depicts DERs as autonomous agents and provides centralized training and distributed execution. In the proposed model, DERs scheduling does not need to constantly connect to a centralized server in the distributed execution phase, which improves the system's robustness and provides data privacy for DER owners. This is also a significant distinction from existing studies.

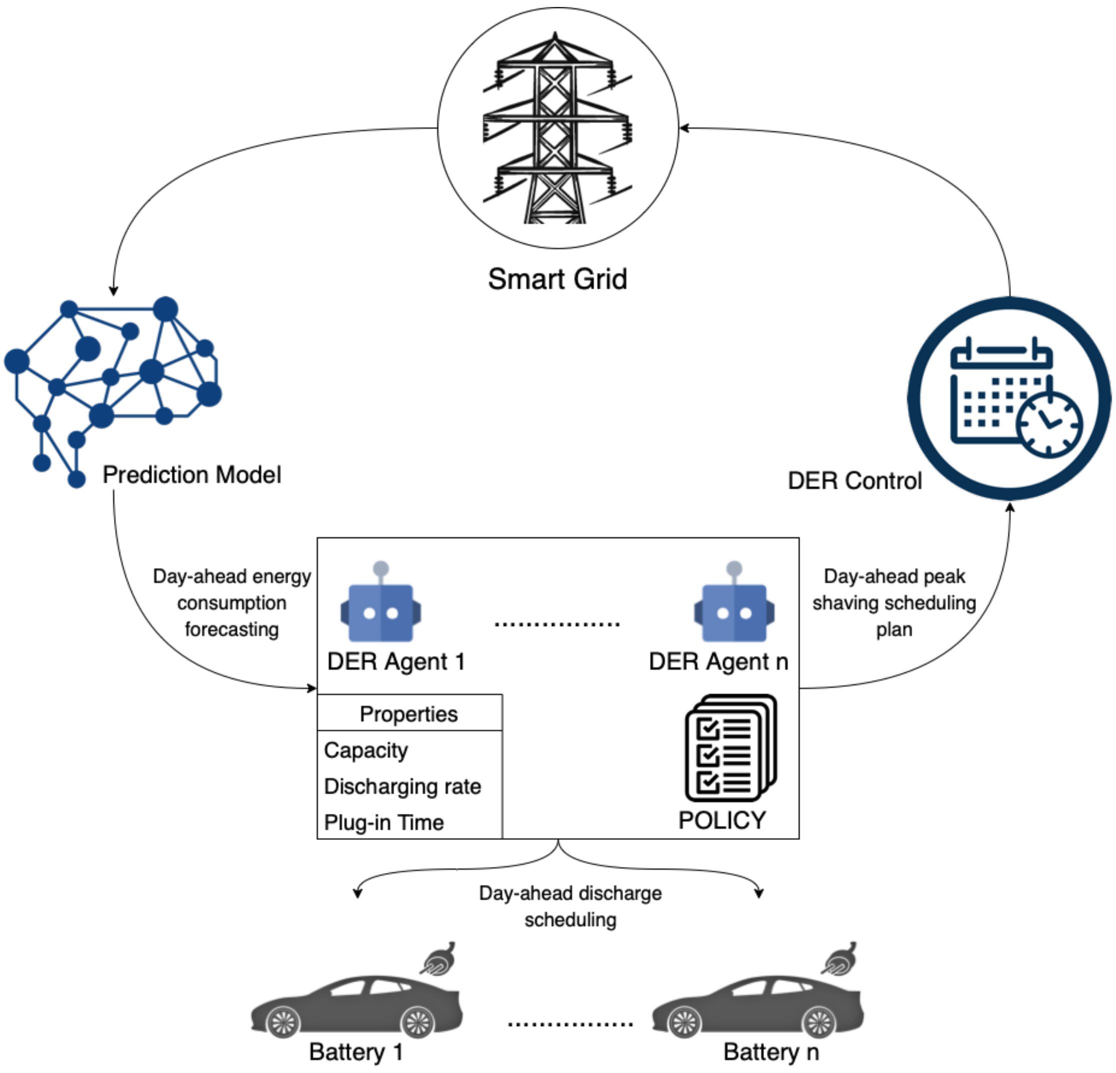


FIGURE 4.1: System Model of Day-ahead Discharging Scheduling

4.2 System model

This section presents the details of the proposed model and its design components. Figure (4.1) shows the system architecture. In our system, each DER (e.g., an EV battery) is modelled as an agent. The agents are part of a MARL model that determines a day-ahead discharging schedule for multiple DERs. We assume that the MARL model takes its energy forecast from the utility company (electricity provider) through a prediction model designed for this purpose. Once the prediction information is passed to the MARL, the system then coordinates the discharging process. Each agent uses the day-ahead energy consumption prediction to know its discharging schedule based on its properties and policy. The agents' policy is trained in a centralized fashion by a neural network model using an actor-critic algorithm. The proposed system encompasses two

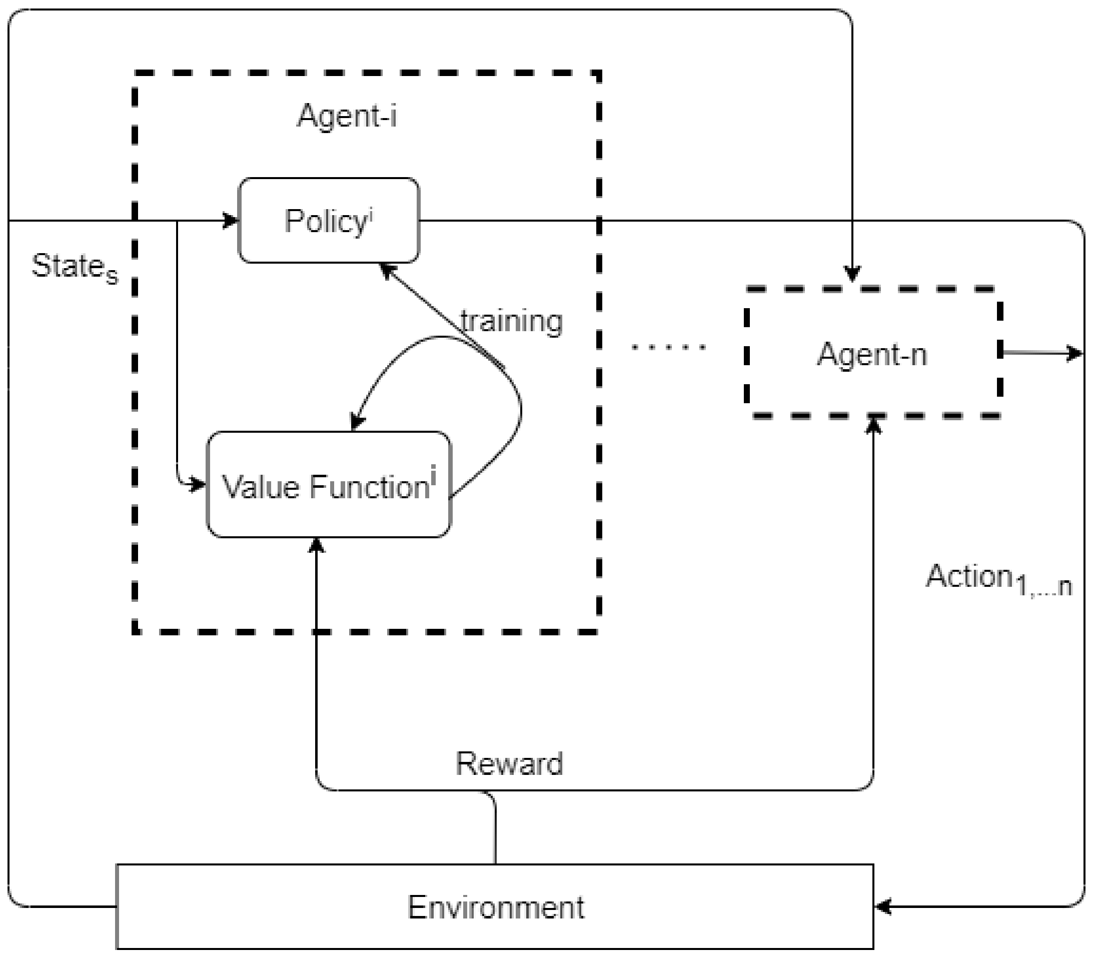


FIGURE 4.2: The training process of actor-critic multi agent reinforcement learning

neural networks: the critic network, which estimates the reward of each agent's discharging schedule (i.e., the action-value), and the actor network, which updates the policy based on the suggestion of the critic network. Figure (4.2) describes the training process. For each training day $d-1$, the energy provider forecasts the power consumption of day d. The policy neural network of each agent i takes state d as input and produces the action a^t, which is the discharging start time for the day d. The energy provider collects all actions from agents and calculates the reward, which is the profit of energy arbitrage and sends it to each agent. The agent then updates the value function based on the reward. The value function is another neural network that takes the state and action as inputs to estimate the reward of each action. Finally, the policy network updates itself based on the action's estimated reward by the value network. The detailed components of the proposed model are described below.

4.2.1 Environment Design

Let $\mathcal{N} = \{1, 2, ..., n\}$ be the set of agents comprising EV batteries who are participating in the DR program of the smart grid system. We assume that each agent $i \in \mathcal{N}$ purchases electricity from the grid during the low peak period at a price rate λ_l. In each day d, the SGP submits a daily power consumption prediction array in 5-min interval EP_d to the agents. The system intelligently selects a discharging start time t_i for agent i and continually discharging at a rate p_i. Each agent has a discharging threshold k_i that defines the maximum amount of energy that can be given back to the grid. The threshold can be determined based on the next travel plan or other usages of the EV. The scheme of determining the threshold is not considered in this article, but it is assumed to be known based on the work in [72][73]. We also assume that each agent can give back to the grid any amount c_i such that $c_i \leq k_i$. The discharging period of agent i in a day is as follows.

$$\tau_d^i = [t : t + \frac{c_i}{p_i}] \tag{4.1}$$

Equation (4.1) indicates that each agent starts discharging at a specified time for a duration $t + \frac{c_i}{p_i}$. The duration of the discharging may not lead to discharging energy more than the defined threshold. However, the agent may be involved in discharging the EV battery more than one time. We define x_i to be the number of times (duration τ_i) that an agent participates in the DR signal. However, it is recommended that an agent participate once a day during the highest peak to save the EV's battery lifetime. At the end of each day, the grid calculates the profit of agent i based on the amount of energy it discharged as shown in equation (4.2).

$$Profit_i^x = \sum_{m=1}^{x} [\lambda_h^m - \lambda_l^m] c_i^m \tag{4.2}$$

Equation (4.2) calculates the profit resulting from charging at a low price of electricity and discharging at a high price of electricity. Each agent would be able to gain a monetary reward based on the amount of energy given to the grid.

4.2.2 Markov Decision Process

The design of the MDP framework is crucial in MARL. Mathematically, our Markov game is defined as a tuple $(\mathcal{N}, S, A, R)$ where $\mathcal{N}$ is the set of agents; S is a finite state space; A is a joint action space of the agents; R is the immediate reward for the agents. In our Markov game framework, each day is considered one state of specific prediction of energy consumption of kWh array of 5-min intervals $t \in d$ given by ep_d^t. Therefore, we have 288 5-min intervals of energy consumption data each day. The discrete action space is $A = [a^t, a^{t+1}, ..., a^T]$ spans the interval of times from $t = 1$ to $T = 288$, and a^t represents the action of agent i to discharge or not from the EV battery into the grid starting at time t. The individual reward for each agent $r_d^i = Profit_i^x$; in our design, we used a fully cooperative setting that all agents share a common reward function as in equation (4.3).

$$R_d = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} r_d^i \tag{4.3}$$

Under some electricity price plans, the price is only affected by the highest peak. In this scenario, R_d is mathematically equivalent to the peak shave, the difference between the maximum energy consumption before and after batteries discharging. We use a shared reward function to avoid competition among the agents. In a non-cooperative setting, the overall system may not maximize the peak-shaving of the smart grid; thus we use a shared reward for each agent to train coordinated policies.

4.2.3 Algorithm Learning Design

Let $\pi^{i*}(a|s)$ be the theoretical optimal policy function of the probability of agent i selecting action a in state s. The policy π^{i*} takes state s as input and output a probability space of actions $\pi^{i*}(s) = [Pr(a^t), Pr(a^{t+1}), ..., Pr(a^T)]$, $\sum_{a \in A} \pi^{i*}(a|s) = 1$. The action-value function in equation (4.4) describes the expected reward of agent i doing action a^t at state $s \in S$. The state-value function in equation (4.5) describes the expected reward of state s by applying the joint optimal policy π^*.

$$Q_\pi^i(s, a) = \mathbb{E}[R_d | S = s, A = a^t] \tag{4.4}$$

$$V_\pi^{i*}(s) = \sum_{a \in A} \pi^{i*}(a|s) \cdot Q_\pi^i(s, a) \tag{4.5}$$

We use one neural network $\pi^i(a|s, \theta^i)$ (actor) to approximate the optimal policy function $\pi^{i*}(a|s)$ and one neural network $q^i(s, a; w^i)$ (critic) to approximate the action-value function $Q_\pi^i(s, a)$; where θ and w are trainable parameters. The approximate state-value function then become equation (4.6). The goal of learning parameters is to optimize objective function $J(\theta)$ in equation (4.7) of the joint policy parameters $\theta^i \in \{\theta^{1,2,...,\mathcal{N}}\}$.

$$V^i(s, \theta^i, w^i) = \sum_{a \in A} \pi^i(a|s, \theta^i) \cdot q^i(s, a; w^i) \tag{4.6}$$

$$J(\theta) = \sum_{s \in S} \sum_{i=1}^{\mathcal{N}} V_{\pi\theta}^{i} \tag{4.7}$$

In the training phase, we have a number of days in the training set, and the aim is to optimize $J(\theta)$. The training phase is shown in Algorithm (2), which is described next.

At the start of the training phase, for each agent i, we configure a different learning rate. Learning rate is a hyper-parameter in deep learning representing the amount of weight updated during training. A default value for the learning rate is 0.1 or 0.01 as a start point to configure. Different neural networks with different layers and neurons have different optimal learning rates, which are challenging and time-consuming to find. Considering learning rate α and β for value neural network and policy neural network respectively, we have $\alpha^i = \delta^i \alpha$ and $\beta^i = \delta^i \beta$. The δ^i is calculated as $\delta^i = \frac{\sum_{i=1}^{n} Ci}{nC_i}$. Each neural network of a different agent will have a different learning rate. The agents with larger battery capacity have a lower learning rate. The agent with a larger battery capacity has a larger effect on the system; updating its policy slower than the agent has less effect during the training.

The next step is initializing the parameters of the neural network θ^i and w^i for each agent. Then initializing the minimum exploration rate and exploration decay ε. The exploration rate is initialized as 1, and at each epoch exploration rate will decrease by the decay ε and stop at the minimum exploration rate, which is typically set to a default value of 0.01. The exploration rate is the chance of the agent performing a random action instead of choosing an effort based on the current model. The details of the effect of exploration rate and decay are described in section (4.4.3).

In figure (4.2), for each training day of each epoch, every agent first observes the state. The observation of the state means reading the predicted power consumption EP^d as the input to each agent's neural network. Then each agent chooses an action based on the current exploration rate. When every agent chooses an action, we can calculate the joint reward R_d. Every value neural network in section (4.2.4) and policy neural network in section (4.2.5) will update their parameter θ^i and w^i base on R_d.

4.2.4 Value neural network

In the model design, we use the actor-critic method [18]. Actor and critic are synonyms for the policy and value function, respectively. The actor-critic method is a preferred algorithm in solving RL problems, which have advantages over both actor-only and critic-only method.

Algorithm 2 Training Phase

```
1: Initializing learning rate parameters $\alpha$ and $\beta$
2: Initializing $\theta^i$ and $w^i$ for each agent
3: Initializing minimum exploring rate and epsilon decay $\varepsilon$
4: for each epoch do
5:     for each day $d$ in the training set do
6:         for each agent $i \in \mathcal{N}$ do
7:             Observe the state $s$
8:             Randomly sample action $a$ according to $\pi^i(s, \theta^i)$ and exploration rate
9:             Perform action $a$ for each agent
10:        end for
11:        Calculate the reward $R_d$ based on joint actions
12:        for each agent $i \in \mathcal{N}$ do
13:            update $w^i$ using temporal difference in section (4.2.4)
14:            update $\theta^i$ using policy gradient in section (4.2.5)
15:        end for
16:    end for
17: end for
```

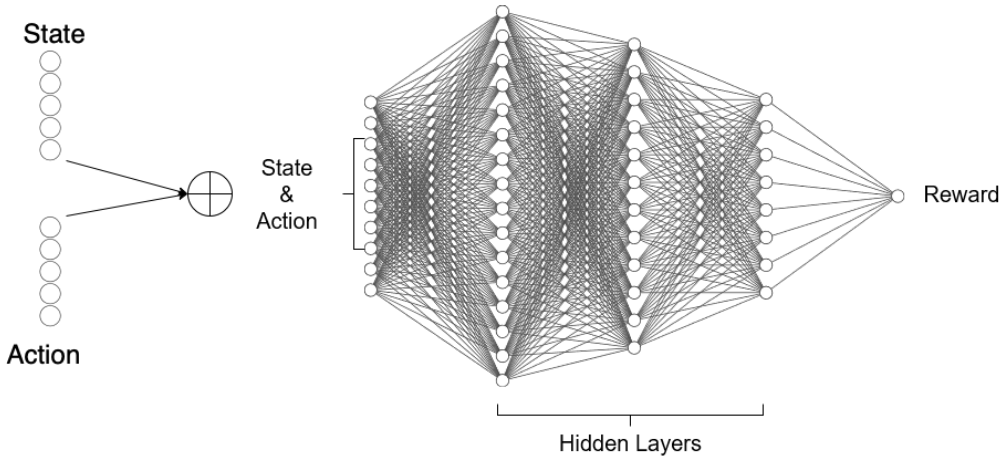


FIGURE 4.3: Value Neural Network(Critic)

The value neural network, as shown in figure (4.3), takes the merged array of state and action space as input. The state is the daily power consumption EP^d. The action is discrete, so the action array is generated by one-hot encoding. After the hidden layers, the value neural network uses a dense layer to output one value representing reward. This output predicts the expected reward of input action in the input state. The training process of updating trainable parameter w aims to increase the precision of the value neural network output to better estimate R_d.

At the end of each training case, the value neural network updates itself using the temporal difference (TD) method. The TD target is the actual reward R_d. The loss function in equation (4.8) encourage to minimize the difference of the value neural network $q(s, a; w)$

and actual reward R_d. In the process of gradient descent in equation (4.9), w will update based on the derivative of loss function to w.(α is the learning rate) After gradient descent, the loss $L(w)$ will decrease thus the output of value neural network will closer to R_d.

Loss function:

$$L(w) = \frac{1}{2}[q(s, a; w) - R_d]^2 \tag{4.8}$$

Gradient descent:

$$w_{d+1} = w_d - \alpha \cdot \frac{\partial L(w)}{\partial w} \tag{4.9}$$

4.2.5 Policy neural network

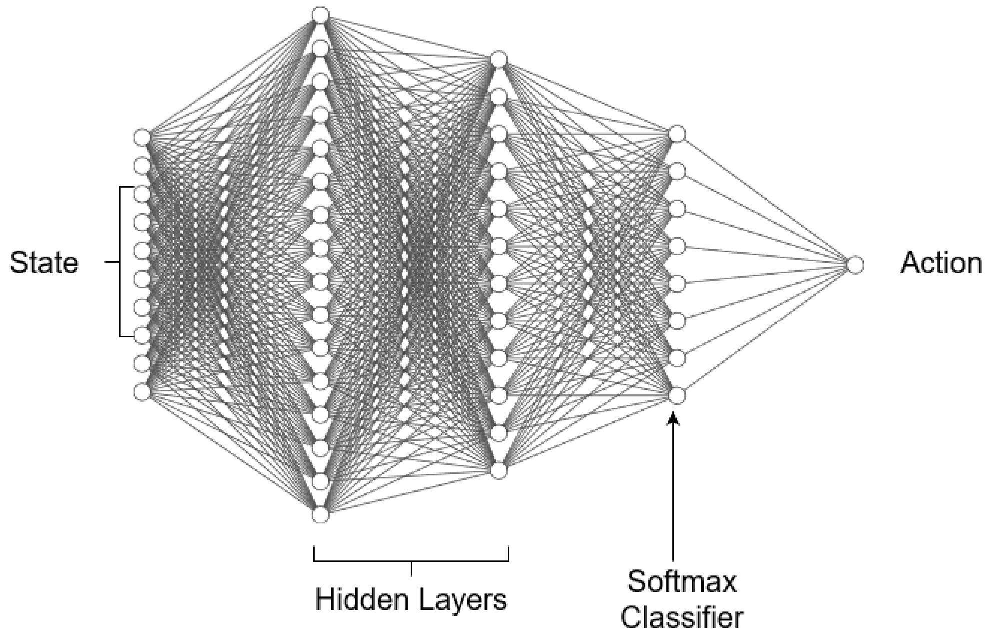


FIGURE 4.4: Policy Neural Network

The policy neural network, as shown in figure (4.4), takes only the state as the input feature. After the hidden layer, the policy neural network uses a dense layer map to vector the same size as action space. Then the softmax function will output a probability of suggested actions. The final output action could be the highest possible action in the previous layer. Updating the policy neural network aims to gain a higher reward from the output suggested action.

The policy neural network is updated using the policy gradient. The state value functions $V(s; \theta, w)$ output the expected reward, as the performance, of policy function in state s. In the process of gradient ascent described in equation (4.13), the trainable parameter θ is adding the derivative of state-value function $g(a, \theta)$.(β is the learning rate) The updated policy network will increase the value of state-value function $V(s;\theta,w)$ which increase the performance of the policy network. The derivative of the state-value function shown in equation (4.12), $q(s,a;w)$ is the value neural network. The equation (4.10),(4.11) derives why log is used in $g(a,\theta)$. Derivative of state-value function $V(s;\theta,w)$:

$$g(a,\theta) = \sum_{a\in A} \frac{\partial \pi(a|s;\theta)}{\partial \theta} \cdot q(s,a;w) \tag{4.10}$$

$$g(a,\theta) = \sum_{a\in A} \pi(a|s;\theta) \frac{\partial \pi(a|s;\theta))}{\partial \theta \cdot \pi(a|s;\theta)} \cdot q(s,a;w) \tag{4.11}$$

$$g(a,\theta) = \frac{\partial \log \pi(a|s;\theta))}{\partial \theta} \cdot q(s,a;w) \tag{4.12}$$

Gradient ascent:

$$\theta_{d+1} = \theta_d + \beta g(a,\theta_d) \tag{4.13}$$

.

4.3 Experiment and Results

The following section introduces the experiments on the performance of the model. The daily energy consumption data are collected from the local utility company Synergy North, Thunder Bay, Ontario, Canada. The prediction model with the historical daily forecasted data is present in [42]. The configuration of the experiments is as follows.

Epoch configuration: In each epoch, the models are trained for all the days in the training sets. The default value for the number of epochs is set to 1000.

Agent configurations: In the initial attempt, we trained three agents with similar battery capacity and discharging rates. Each agent has a 2kW discharging rate and 1 hour discharging period. The training set typically has 50kW power consumption for peak points. The agents were trained to choose a discharging start time from 11:00 am to 10:55 pm in 5-min intervals. Thus the action space contains 144 discrete actions. The scheduling of the discharging considers only the afternoon peak and assumes that EV batteries are available during that time.

Measure of success: The total reward, i.e., the sum of the joint rewards of every agent in all training days.

Average Peak Shave: For each training day, after the models perform discharging activities, the peak reduction is calculated by the highest peak of the actual load curve and after shaved load curve. Average peak shave is the mean of peak shaves through all training days.

Neural network: The policy neural network has three layers. Layer1 includes 288 input space to read a 5-min interval power consumption of training day, 512 neurons. Layer 2 includes 256 neurons. Layer 3 includes a dense layer of 144 action space. The value neural network has the same hidden layer as the actor neural network. The value neural network input layer includes a 5-min interval of power consumption merged with the 144 action space. The dense layer is reduced to one number as the reward for training the actor neural network.

4.3.1 Experiment on shared reward function

The experiment focuses on the coordination of multiple agents to perform peak shaving. The reward learned by the model for this experiment is the joint profit of the three agent's activities. Each agent gets a profit as described in equation (4.2) from its discharging activities. The total reward is shown in figure (4.5), and the average peak shave is shown in figure (4.6). The result of the trained model for an example day in the validation set is shown in figure (4.7). The results of the experiment prove that the models optimize the discharging scheduling and the trained model provides optimal results with coordination. The reward figure demonstrates that the model is learning based on the designed reward function; the peak shave figure shows the model's enhancement during the training phase optimized agents' discharging policy. The agent's trained model does not require other agents' information, and thus they are distributed executable.

4.3.2 Experiments on scheduling both charging and discharging

The discharging scheduling is the most important in energy arbitrage. Currently, these DERs charges are performed at night when there is less demand. But this can also pose a problem with the increase in DER and hence, create demand if the charging is performed simultaneously. The proposed model can take care of that. The model generates the charging and discharging schedules for each DER. Figure (4.8) shows the model predictions for charging and discharging to maintain a stable load. The experiment is training five charging scheduling models with the same power and battery capacity as discharging devices. The reward learned by the charging scheduling model is negative to the discharging models. The observed results prove that the system can effectively generate charging and discharging schedules.

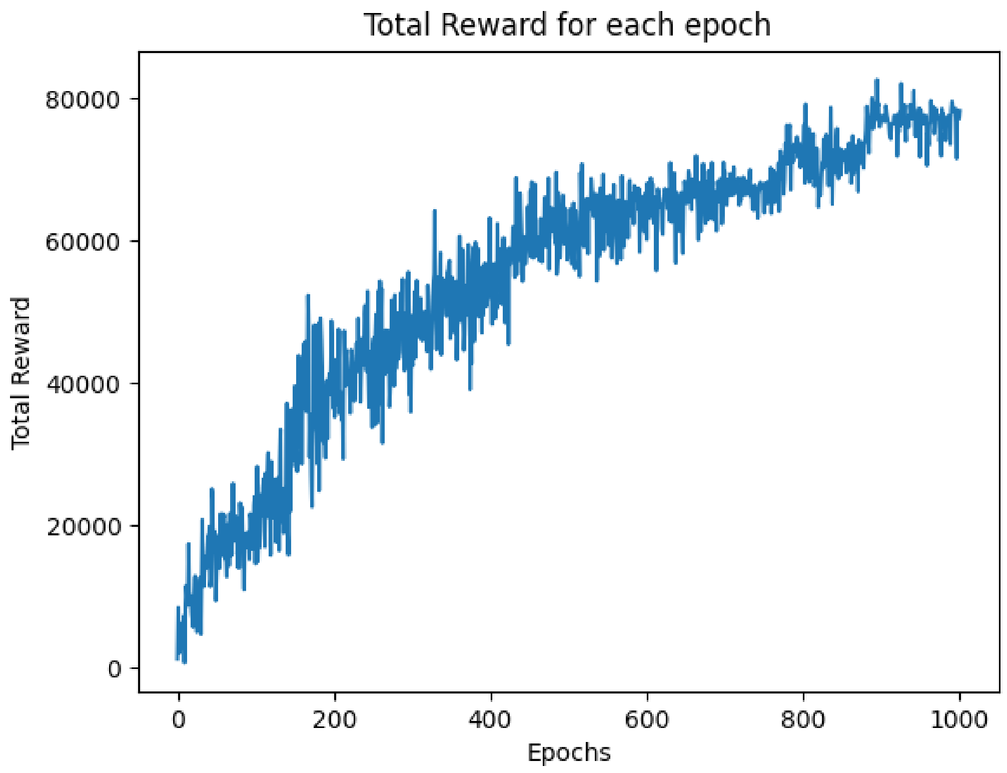


FIGURE 4.5: Reward for each training epoch of Experiment on shared reward function

4.4 Time Complexity Experiment

The time complexity analysis of artificial intelligence models differs from traditional algorithms. The time complexity of an artificial intelligence model has two components: training time and inference time. Training time is the time required to train a model in order to achieve desired results. Inference time is the time taken by the model to process the input and produce an output. The time complexity analysis of artificial intelligence algorithms is less critical than traditional ones since the training phase's time difference does not affect the actual application. Training time is subjective to many parameters, for example, training data size, model complexity, and desired results. However, it is still meaningful to analyze the time complexity of artificial intelligence algorithms. The training dataset usually contains a large amount of data; the time complexity of artificial intelligence should be less than $O(n^2)$ to avoid exponential time growth. In the time complexity analysis, we focus on the relationship of actual time taken in the training phase. The time of each experiment is recorded on the same machine with Intel(R) Core(TM) i7-6700 CPU 3.40 GHz, 16GB RAM, and Radeon (TM) RX 480 GPU. The objective of the following experiment is to prove that the model's training time will not

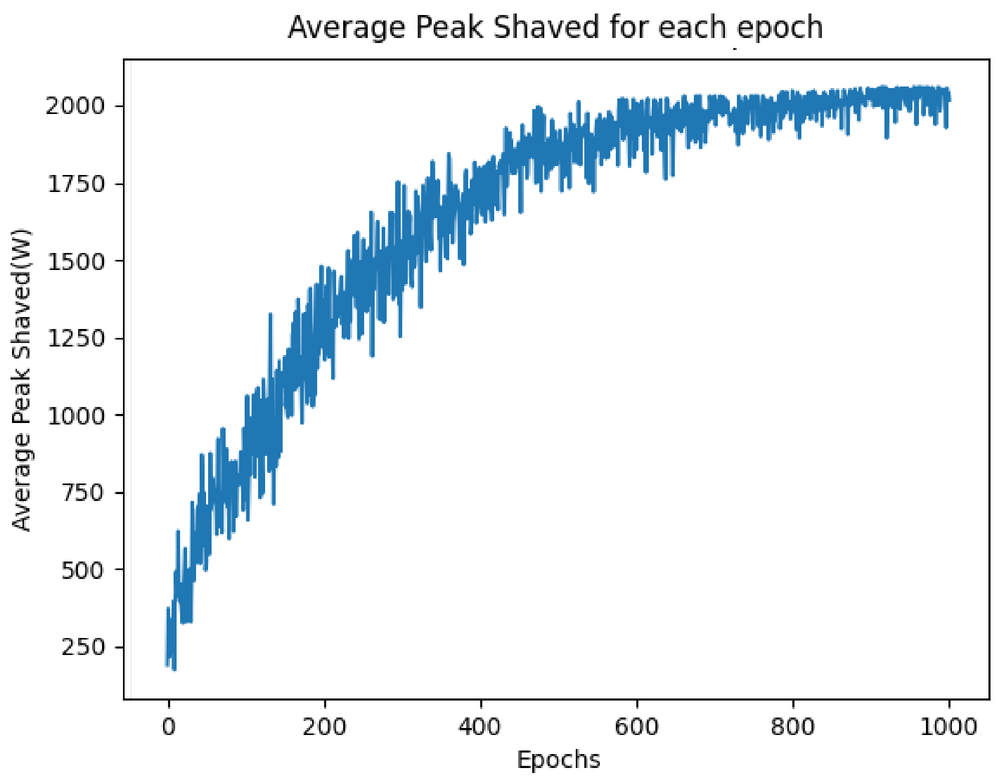


FIGURE 4.6: Peak Shave for each training epoch of Experiment on shared reward function

grow exponentially with linearly increasing training data or other possible changes in the model structure.

4.4.1 Training Time and Agent Sizes

The objective of this experiment is to analyze the relationship between training time and agent size. The agent size represents the number of DERs in the system. Figure (4.9) exhibits a linear relationship between the training time and the size of the agents in the model. Hence, we can conclude that the time complexity is $O(n)$.

4.4.2 Training Time and Action space

This experiment analyzes the relationship between training time and the resolution of action space. The discharge recommendation system provides an action to agents. The action is allocated from the provided action space. The action space in our model is the available discharging times. The resolution of action space decides the performance of

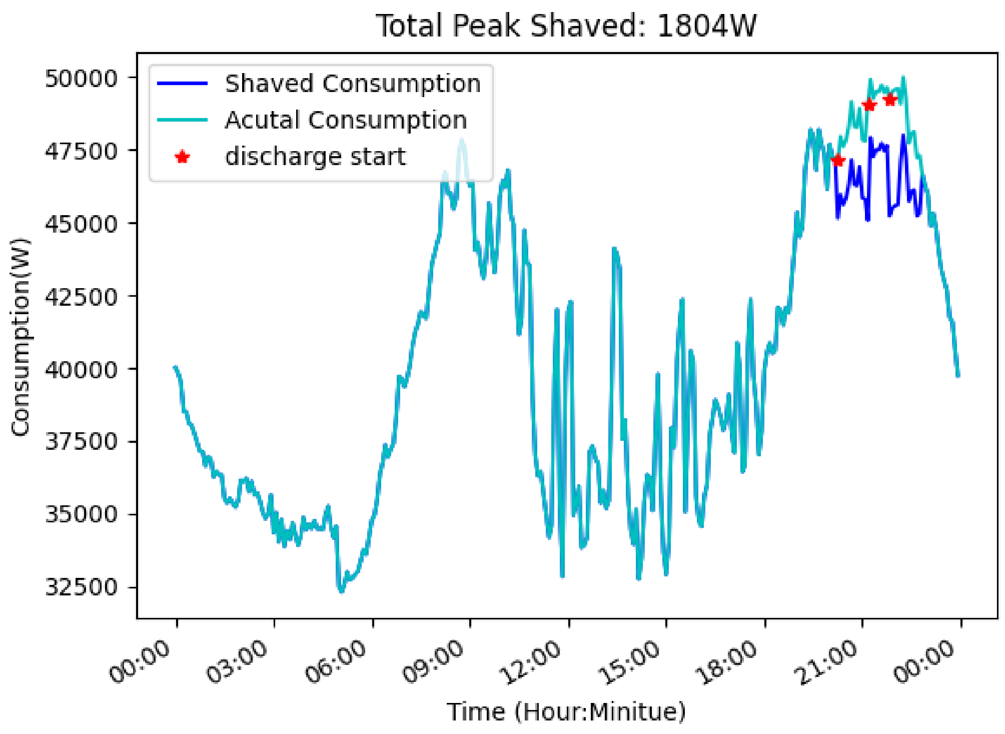


FIGURE 4.7: Example day in validation set

discharging effectiveness of the system. The higher the resolution is, the granular the action space, and hence, the system could provide more precise discharging scheduling. For example, if the resolution is one hour, the system could only suggest DER discharge on the hour (such as 15:00), and the action space will be 24 for a day. If the resolution is five minutes, the system could suggest DER discharge at each five-minute interval (such as 15:05), and the action space will be 288 actions for a day. Due to the limitation of the recorded power consumption data from the smart grid provider, the resolution space is always discrete. In our experiment, the recorded power consumption data is in five minutes intervals. Therefore, we tested from five minutes to one hour of action space resolution. Figure (4.10) shows that there is a linear relationship between the action space and training time. Based on our results, we can conclude that the trade-off between the action space and training time is not significant. Therefore, a larger action space should be used for higher precision. That is because the action space resolution change only affects the dense layer of the neural network model. There are still immense calculations needed for the hidden layers. In conclusion, only decreasing the resolution of action space is not optimal for speeding up the training time.

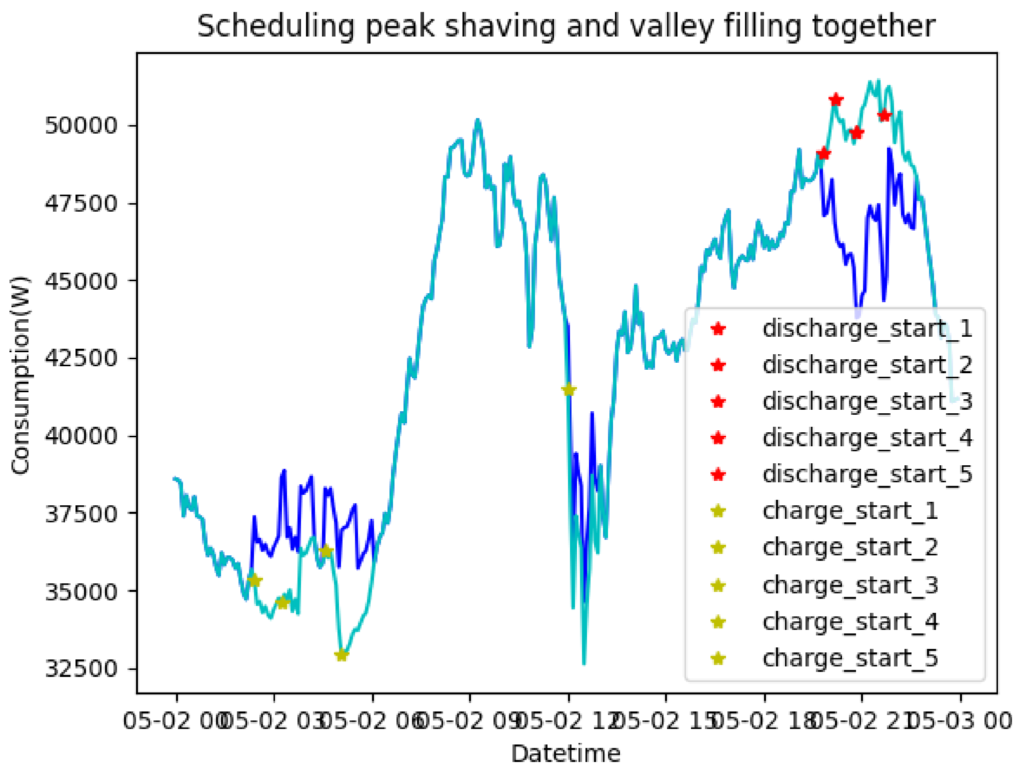


FIGURE 4.8: Example day of both discharging and charging scheduling

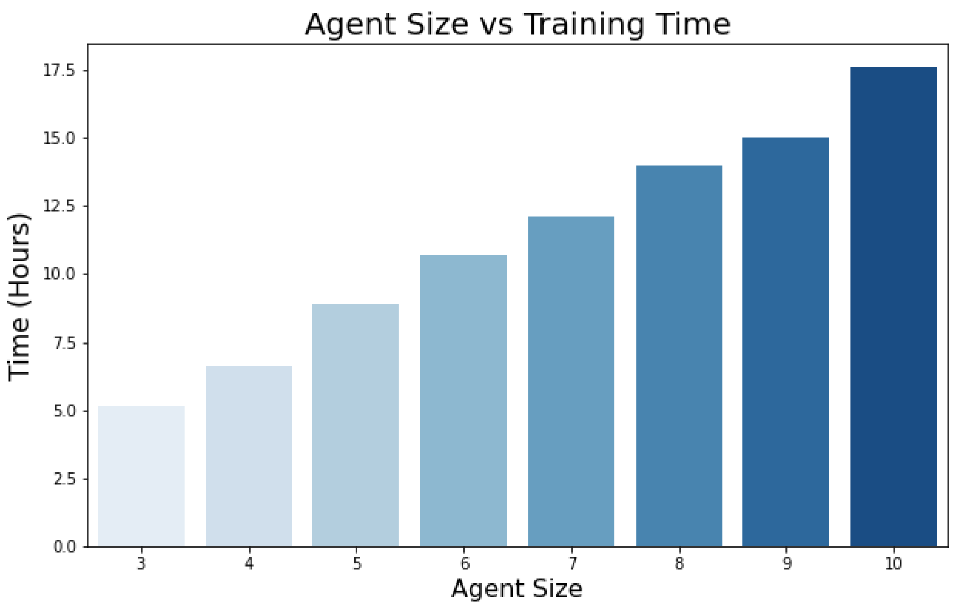


FIGURE 4.9: Relationship between Training time and Agent size

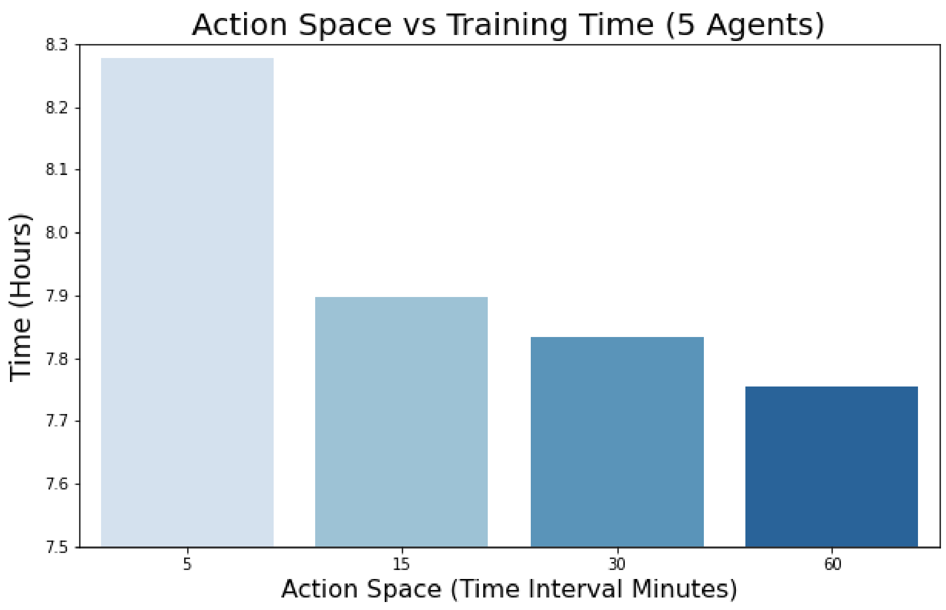


FIGURE 4.10: Relationship between training time and action space

4.4.3 Effect of Exploration rates

To achieve the desired goals, reinforcement learning agents use two methods: Exploration and Exploitation. In the exploration phase, the agent explores new actions and the rewards for a given state. Whereas in the exploitation phase, the agent makes decisions using the previous knowledge. For an agent to maximize the reward function, it performs heavy exploration in the initial iterations for the agent to explore the best actions at every given stage and to achieve the maximum reward. The exploration rate represents the probability of our agent exploring the environment. During the training phase of the agents, the exploration rate is initialized as 1 to perform exploration. To use the best actions explored in the initial iterations, the exploration rate is reduced by a decay function ε. Therefore over time, the model uses exploitation over exploration to maximize the reward.

Figure (4.11) shows the effect of different exploration rates on the model.

Based on our results, we can conclude that using a smaller value of ε can help the model converge earlier and hence useless epochs. The value of ε has to be adjusted based on the action space. Based on our earlier experiment (4.4.2), we can conclude that the training time can further be reduced for a larger action space by choosing an optimal ε value.

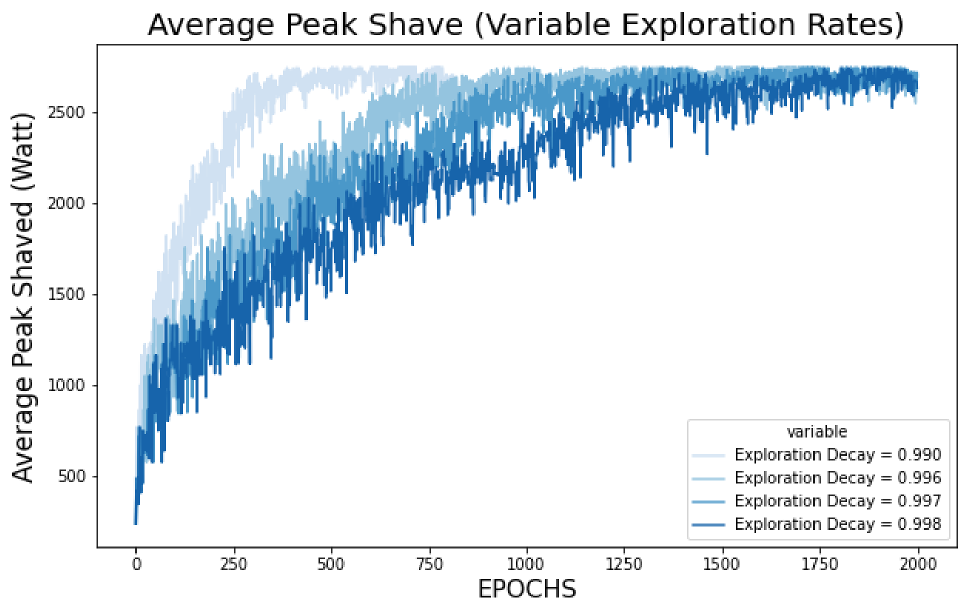


FIGURE 4.11: Average Peak Shave using different Exploration Rates

4.4.3.1 Serial and Parallel Programming

The training and inference of agents can be sped up using parallel programming. Using multi-core to improve program performance usually requires some processing of computationally intensive code. The process includes the following three steps. First, divide the code into blocks. Second, execute these code blocks in parallel through multiple threads. After the results become available, integrate these results in a thread-safe and high-performance manner. There is a challenge that when multiple threads use the same data simultaneously, the common strategy of locking for thread safety considerations will cause competition. As shown in the algorithm (3), model execution and training are executed in parallel. In each simulated day, the random sampling action for each agent are isolated from each other; however, the calculation of reward need to wait until every agent selects its action. The model's training for each agent's after-action reward is calculated and can also be executed parallelly.

Figure (4.12) presents a comparison of training time between serial programming and parallel programming. The application of parallel programming can speed up the training phase remarkably.

Algorithm 3 Training Phase with parallel programming

```
 1: Initializing learning rate parameters $\alpha$ and $\beta$
 2: Initializing $\theta^i$ and $w^i$ for each agent
 3: Initializing minimum exploring rate and epsilon decay $\varepsilon$
 4: for each epoch do
 5:     for each day $d$ in the training set do
 6:         for each agent $i \in \mathcal{N}$ do
 7:             Observe the state $s$                                        }
 8:             Randomly sample action $a$ according to $\pi^i(s, \theta^i)$  } in parallel
 9:             Perform $a$ for each agent                                   }
10:         end for
11:         Wait until all parallel process done
12:         Calculate the reward $R_d$ based on joint actions
13:         for each agent $i \in \mathcal{N}$ do
14:             update $w^i$ using temporal difference                       } in parallel
15:             update $\theta^i$ using policy gradient                      }
16:         end for
17:         Wait until all models updated
18:     end for
19: end for
```

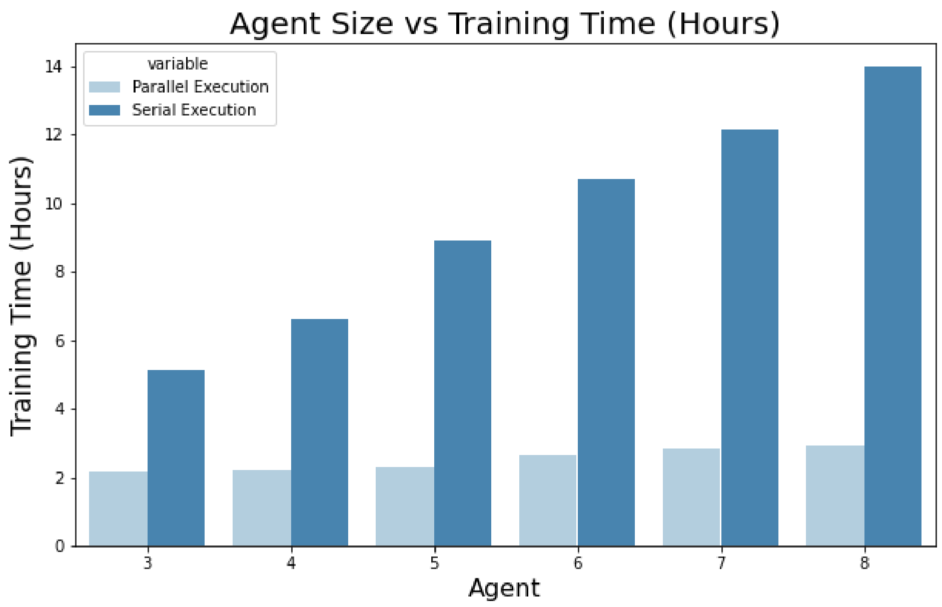


FIGURE 4.12: Comparison of training time between Serial and Parallel programming

4.5 Limitation

Due to the limited variability of the dataset used in the training process, the proposed model fails to detect the peaks occurring in the morning. This happens because the model is trained on a heavily biased dataset where the peaks only occur in the evening and hence only focus on that time frame. Figure (4.13) shows the example in which there are two peaks, morning and evening. Even though the morning peak is more beneficial for shaving, the model chooses the evening peak due to its biased nature. In the future, this problem can be resolved by an assessment mechanism that can help detect the more valuable peak to shave in case of limited DER or utilize the available resources effectively to shave the two peaks instead of opting for the more valuable peak.

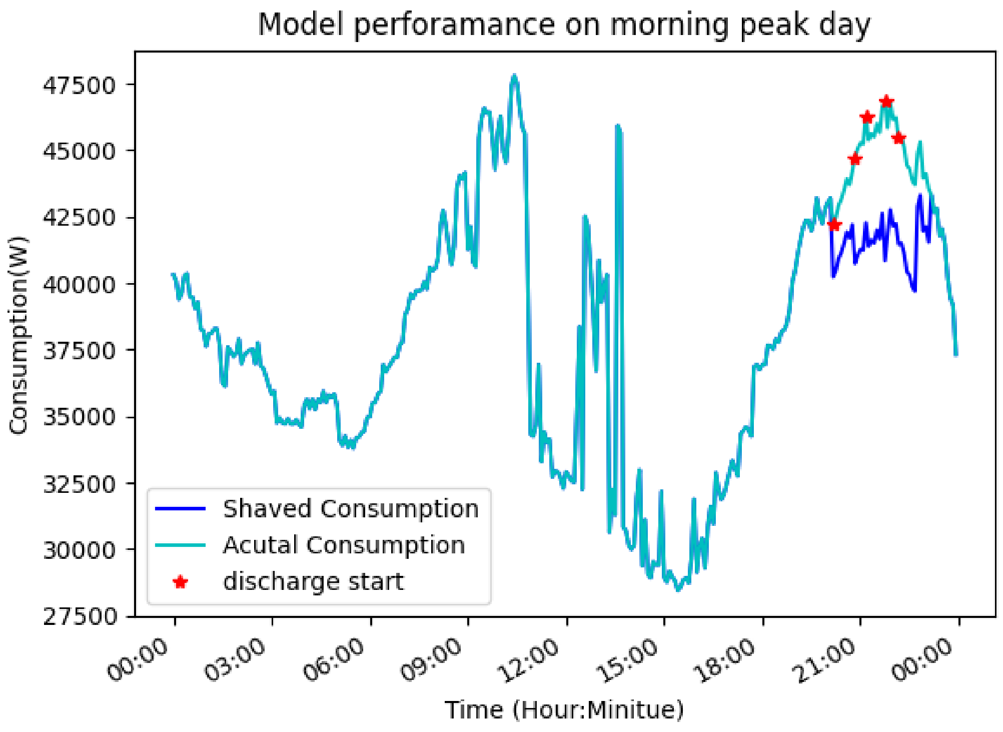


FIGURE 4.13: Example day in validation set : discharging scheduling with highest peak not in available discharging period

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we present an image processing technique to analyze power consumption curves and make enhanced peak predictions during shoulder season. We proved that the pattern of the daily consumption graph in a smaller residential area could be clustered, and days in similar patterns have a stronger correlation. We also proposed a dynamic weighting and majority voting mechanism to improve the ensemble system's performance further. The system enhances the demand response planning during the shoulder season and increases peak load period prediction accuracy. On the other hand, we still need to exterminate if image-based processing could apply to other locations. We also proposed a MARL model for day-ahead DER discharging scheduling to peak shaving the daily consumption curve. The above experiments prove that the MARL model could optimize day-ahead scheduling for DER. The experiments demonstrate that agents could coordinate with each other in the executed phase without communication to maximize the daily peak shaving. Unlike traditional DER charging or discharging scheduler, the distributed execution does not need a collection of discharging times of DER in a centralized server. The DER owner data privacy is highly improved in the proposed system. Data privacy security will increase general customers' interest in enrolling in energy arbitrage since people are becoming increasingly concerned about data privacy. Besides, the distributed execution increases the robustness of networking since each DER agent connects to their local trained model. This thesis also analyzes time complexity that demonstrates that the model training time will not exponentially increase with a massive number of DER installed in the peak shaving scheduling system. The analysis also includes the potential speed-up of training time and the efficient trade-off between training speed and model performance.

5.1.1 Future Work

In this section, we present the future extension of this work with the improvement of model performance.

1. The image processing system uses a pre-trained convolution neural network and k-means clustering. Collecting different datasets will solve the shortage of daily power consumption images. Instead of using transfer learning models, a future direction would be to train a daily consumption pattern classification model with an extensive database. During the database extension, we could unfreeze layers of the current pre-trained model and retrain it with labelled images. Once we get enough variety of labelled daily power consumption images, we could build an optimized image processing model to cluster all types of daily power consumption images. The challenges of this task include the difficulty of collecting other datasets and labelling the daily images. The power consumption data is private for each energy company; data collection requires further cooperation from other companies. The labelled images could be done with the current model or manually labelled by an expert electrical engineer.

2. We also need more experiments to study the effect of the image-based processing technique on the prediction model accuracy. Future work would collect more power consumption data from more geographical locations and data from other non-residential areas such as schools or industries. After collecting extra data, we could analyze if the image-based processing technique is a universal solution for optimizing prediction model performance.

3. Our current work assumes that agents are trained using a centralized approach. A future direction would be to consider federal learning. The application of federal learning will assist the model in creating an average of all existing models for the newly installed agents and keep training distributed during the daily task, eventually becoming an optimal model in the multi-agent system. The application of distributed training phase will highly increase the chances of convincing new DER owners to enroll in the energy arbitrage and future model updating.

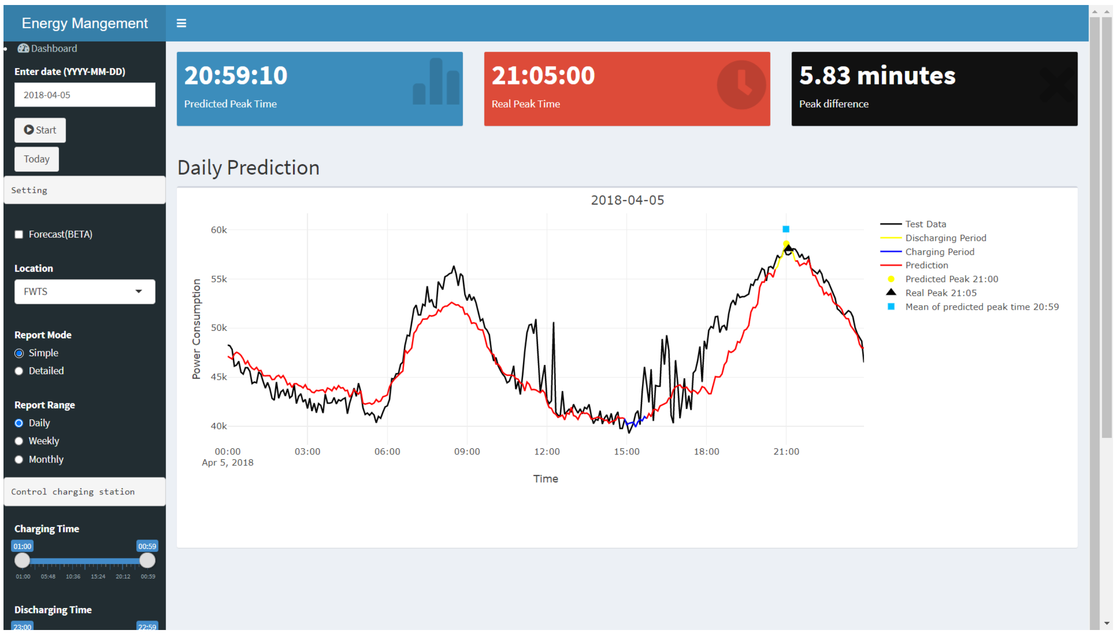
4. Another future work is the assumption that every DER agent will perform a discharging per day. We need future research on the damage of discharging to DER batteries' life. The current system design only schedules one discharging task for all DER per day to save the battery's life. In the future, we could train the MARL models with a reward function considering the damage to the batteries; then, the model can suggest multiple discharging schedules a day with a reasonable calculation of the cost of batteries and save on peak shave. In addition, sometimes, not all

DER discharging will benefit the peak shaving. The increasing number of DERs in the system will lead to part of DERs discharging unnecessary and wasted on certain days. There is a threshold when parts of DER discharging already flatten the peak, then the rest of the discharging activities will become inefficient. The challenge is the design of reward functions needs to consider the cost of batteries discharging financially.

Appendix A

Prototype of Real-life Implementation at Synergy North

The following are the screenshots of the prototype that we have developed and implemented in real-life at the industry partner Synergy North.



Energy Mangement

Dashboard

Enter date (YYYY-MM-DD): 2018-04-05

Start

Today

Setting

Forecast(BETA)

Location: FWTS

Report Mode: [✓ Simple] Detailed

Report Range: [✓ Daily] Weekly Monthly

Control charging station

Charging Time: 01:00 – 00:59

Discharging Time: 23:00 – 22:59

20:59:10 Predicted Peak Time

21:05:00 Real Peak Time

5.83 minutes Peak difference

Daily Prediction

Figure A.1: User interface of simple mode

Figure (A.1) shows the landing page of the system where the user can select the electric feed system, upload the dataset for the specific forecasting day. The result shows the final prediction for the day-ahead and gives the user the predicted highest peak time, the actual peak time, and the difference in time between them.

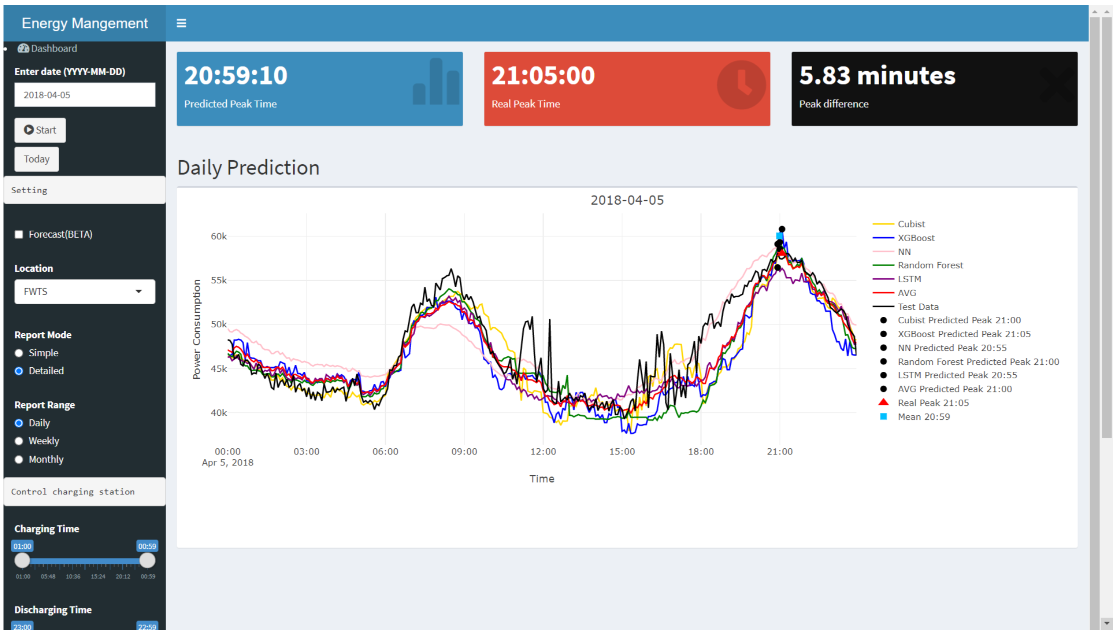


FIGURE A.2: User interface of detailed mode

Figure (A.2) provides the user with details about the prediction models used in the ensemble. The user can try to check the performance of each prediction algorithm in comparison with others.

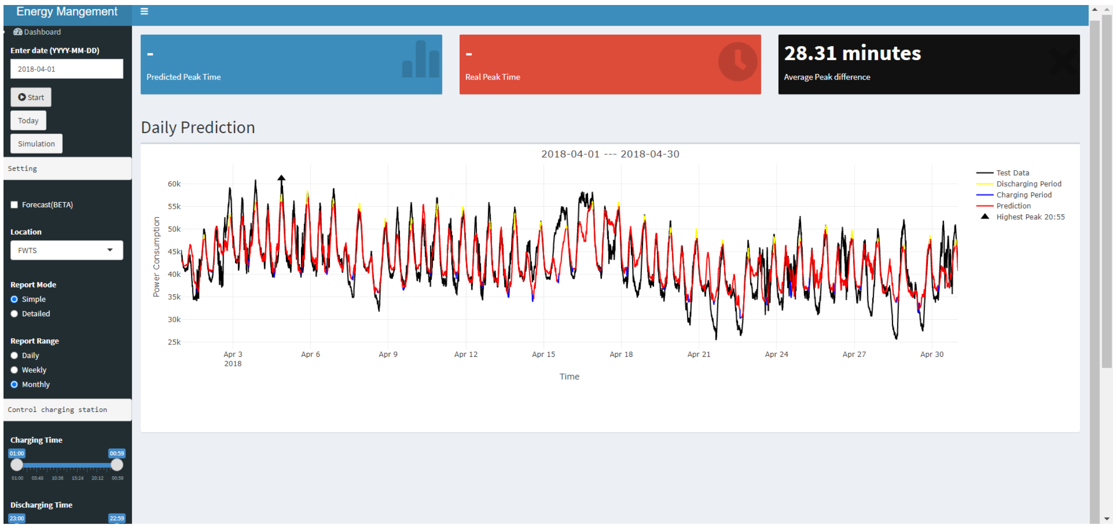


FIGURE A.3: User interface of monthly report simple mode

Figure (A.3) presents the monthly power prediction including a average peak shave. The figure also provide a suggest discharging period and charging period. The Figure (A.4) show the interface for the detailed monthly prediction for each algorithm.

Figure (A.5) is the user interface of the day-ahead scheduling from MARL models. This interface can be accessed from the interface in Figure (A.4). This interface allows the user to enter a date/time, the number of agents they want in the system, and the

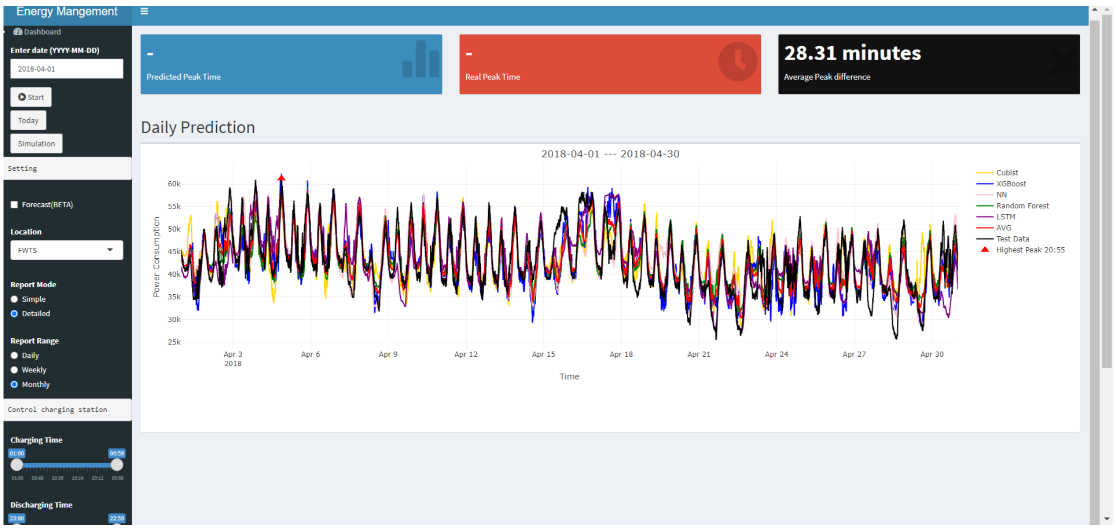


Figure A.4: User interface of monthly report detailed mode

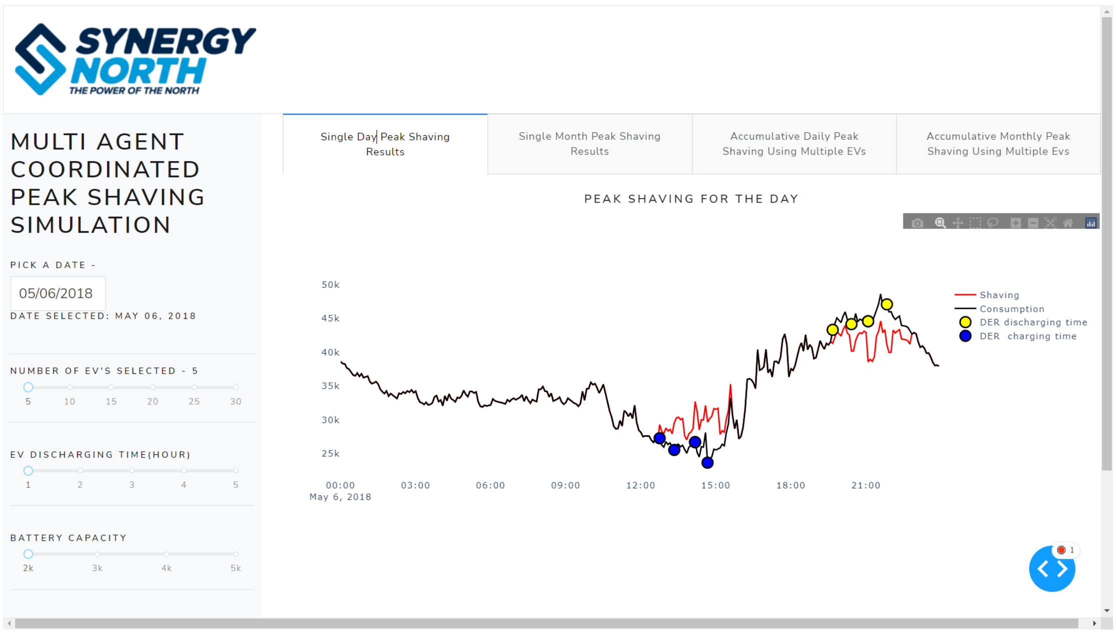
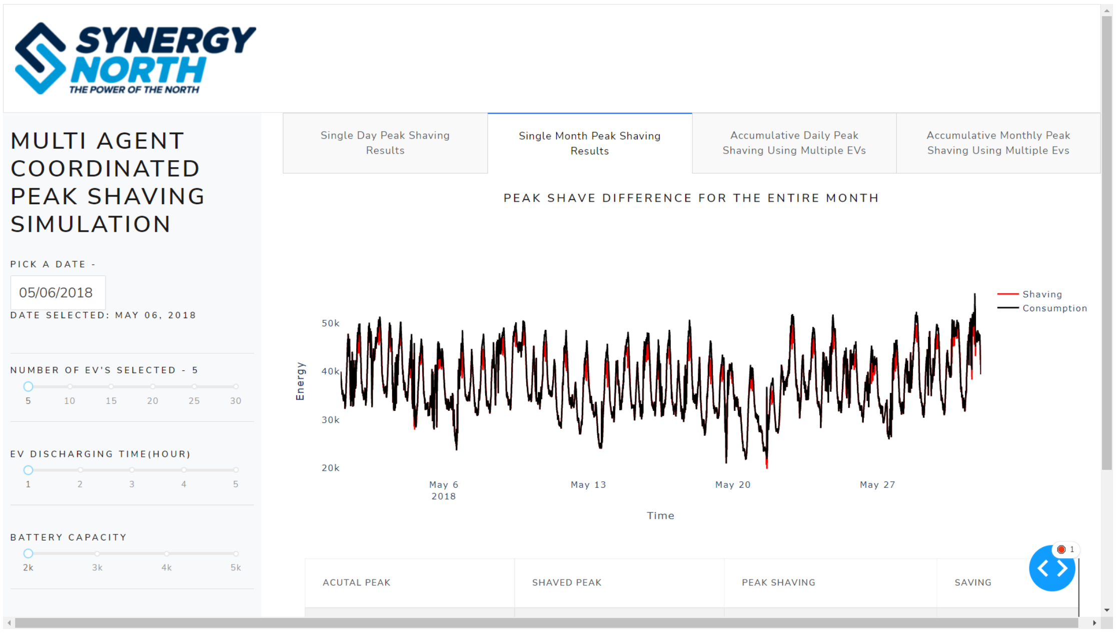

SYNERGY NORTH
THE POWER OF THE NORTH
MULTI AGENT COORDINATED PEAK SHAVING SIMULATION
PICK A DATE -
05/06/2018
DATE SELECTED: MAY 06, 2018
NUMBER OF EV'S SELECTED - 5
EV DISCHARGING TIME(HOUR)
BATTERY CAPACITY
Single Day Peak Shaving Results
Single Month Peak Shaving Results
Accumulative Daily Peak Shaving Using Multiple EVs
Accumulative Monthly Peak Shaving Using Multiple Evs
PEAK SHAVING FOR THE DAY
Shaving
Consumption
DER discharging time
DER charging time
May 6, 2018

Figure A.5: User interface of daily discharging scheduling

configurations of DERs, including discharging time and battery capacity. Figure (A.5) shows the daily discharging simulation.

Figure (A.6) presents the monthly peak shaving results based on the configuration on the left slide bars. The figure updates itself when the user changing the configurations of DERs.

Figure (A.7) is the interface of Accumulative Daily Peak Shaving. This interface allows the user to select agent number, provides a graph of the effect on peak shaving for each agent. Figure (A.8) is a continuation of Figure (A.7) which presents the expected peak shaving increase for each added agent in the system.


SYNERGY NORTH
THE POWER OF THE NORTH
Single Day Peak Shaving Results
Single Month Peak Shaving Results
Accumulative Daily Peak Shaving Using Multiple EVs
Accumulative Monthly Peak Shaving Using Multiple Evs
PEAK SHAVE DIFFERENCE FOR THE ENTIRE MONTH
Shaving
Consumption
Energy
50k
40k
30k
20k
May 6 2018
May 13
May 20
May 27
Time
ACUTAL PEAK
SHAVED PEAK
PEAK SHAVING
SAVING

MULTI AGENT COORDINATED PEAK SHAVING SIMULATION

PICK A DATE -

05/06/2018

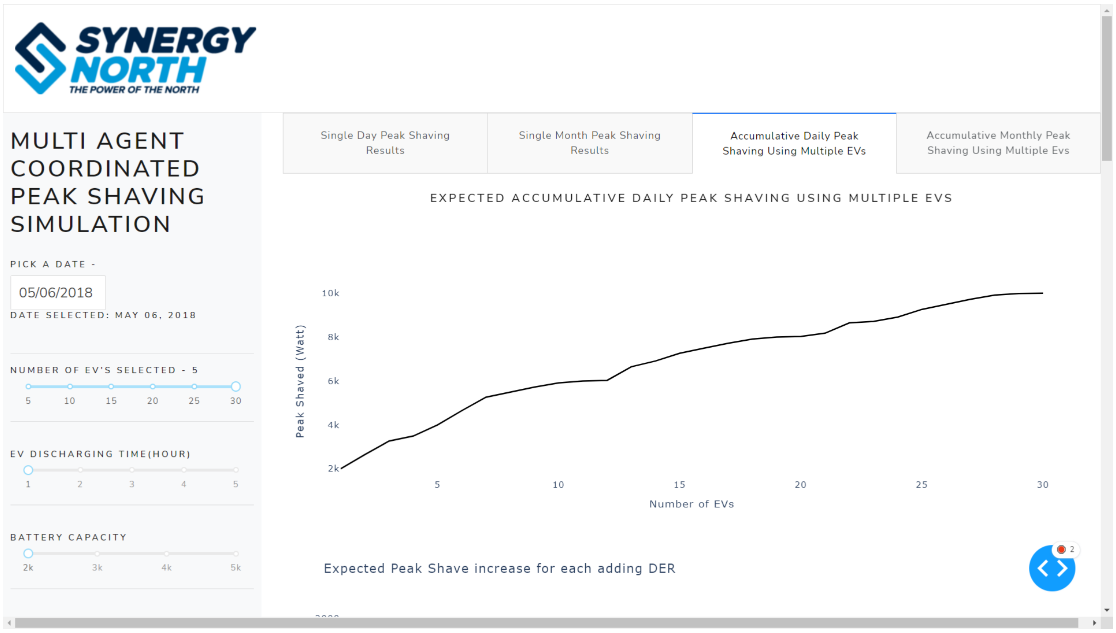
DATE SELECTED: MAY 06, 2018

NUMBER OF EV'S SELECTED - 5

EV DISCHARGING TIME(HOUR)

BATTERY CAPACITY

Figure A.6: User interface of expected monthly peak shaving


SYNERGY NORTH
THE POWER OF THE NORTH
Single Day Peak Shaving Results
Single Month Peak Shaving Results
Accumulative Daily Peak Shaving Using Multiple EVs
Accumulative Monthly Peak Shaving Using Multiple Evs
EXPECTED ACCUMULATIVE DAILY PEAK SHAVING USING MULTIPLE EVS
Peak Shaved (Watt)
10k
8k
6k
4k
2k
5
10
15
20
25
30
Number of EVs

MULTI AGENT COORDINATED PEAK SHAVING SIMULATION

PICK A DATE -

05/06/2018

DATE SELECTED: MAY 06, 2018

NUMBER OF EV'S SELECTED - 5

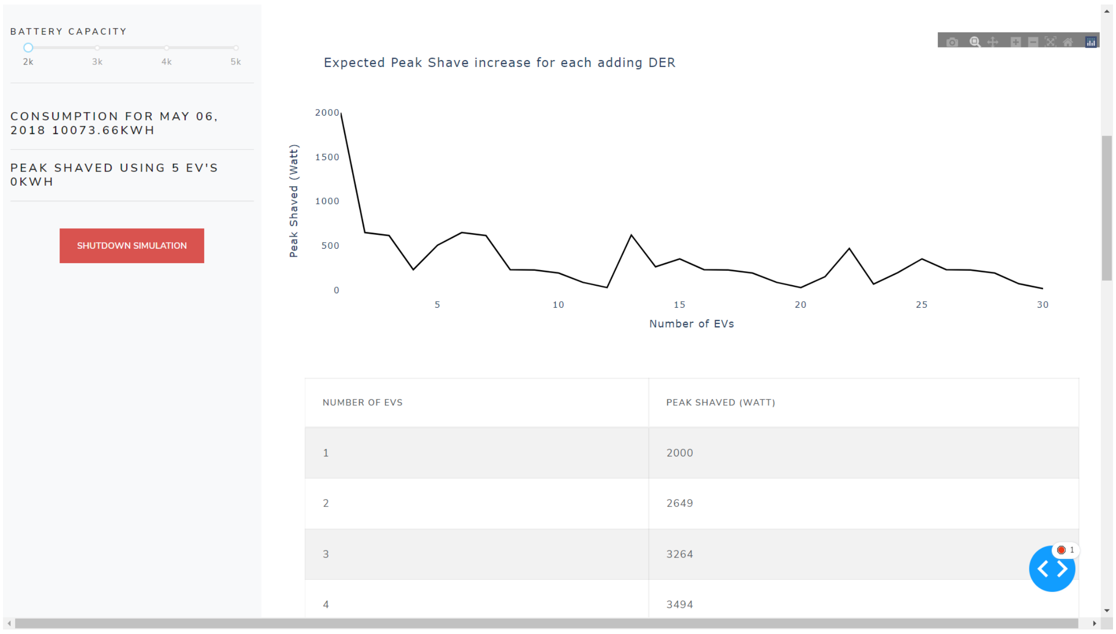
EV DISCHARGING TIME(HOUR)

BATTERY CAPACITY

Expected Peak Shave increase for each adding DER

Figure A.7: User interface of Accumulative Daily Peak Shaving Using Multiple EVs

Figure (A.9) shows the accumulative monthly peak shaving for each added agent. The monthly peak shaving is calculated by the sum of every daily peak shaving.


BATTERY CAPACITY

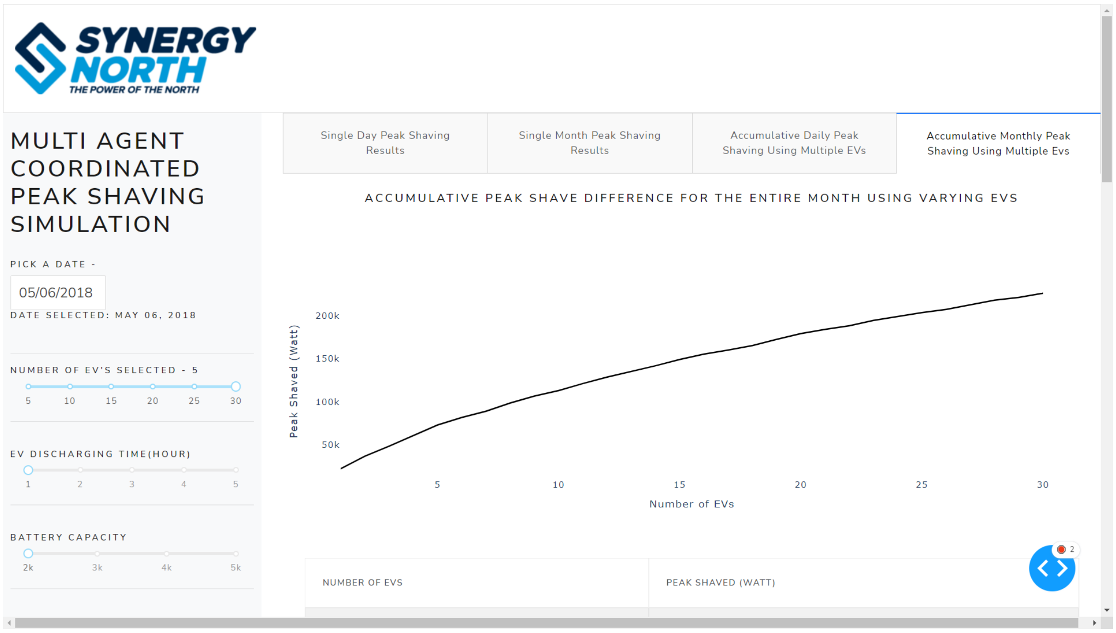
CONSUMPTION FOR MAY 06, 2018 10073.66KWH

PEAK SHAVED USING 5 EV'S 0KWH

SHUTDOWN SIMULATION

NUMBER OF EVS	PEAK SHAVED (WATT)
1	2000
2	2649
3	3264
4	3494

FIGURE A.8: User interface of Accumulative Daily Peak Shaving Using Multiple EVs(continue)


MULTI AGENT COORDINATED PEAK SHAVING SIMULATION

PICK A DATE -

05/06/2018

DATE SELECTED: MAY 06, 2018

NUMBER OF EV'S SELECTED - 5

EV DISCHARGING TIME(HOUR)

BATTERY CAPACITY

Single Day Peak Shaving Results | Single Month Peak Shaving Results | Accumulative Daily Peak Shaving Using Multiple EVs | Accumulative Monthly Peak Shaving Using Multiple Evs

ACCUMULATIVE PEAK SHAVE DIFFERENCE FOR THE ENTIRE MONTH USING VARYING EVS

NUMBER OF EVS	PEAK SHAVED (WATT)

FIGURE A.9: User interface of Accumulative Monthly Peak Shaving Using Multiple EVs

Bibliography

[1] W. Zhong, R. Yu, S. Xie, Y. Zhang and D. K. Y. Yau, "On Stability and Robustness of Demand Response in V2G Mobile Energy Networks," in IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 3203-3212, July 2018.

[2] O. Erdinc, N. G. Paterakis, T. D. P. Mendes, A. G. Bakirtzis and J. P. S. CatalÃ£o, "Smart Household Operation Considering Bi-Directional EV and ESS Utilization by Real-Time Pricing-Based DR," in IEEE Transactions on Smart Grid, vol. 6, no. 3, pp. 1281-1291, May 2015.

[3] K. Mets, T. Verschueren, W. Haerick, C. Develder and F. De Turck, "Optimizing smart energy control strategies for plug-in hybrid electric vehicle charging," 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, Osaka, 2010, pp. 293-299.

[4] Saxena, Harshit, "Forecasting Strategies for Predicting Peak Electric Load Days" (2017). Thesis. Rochester Institute of Technology.

[5] A. Yassine, "Implementation challenges of automatic demand response for households in smart grids" *2016 3rd International Conference on Renewable Energies for Developing Countries (REDEC)*, Zouk Mosbeh, 2016, pp. 1-6.

[6] Krzysztof Gajowniczek and Tomasz Z abkowski. Two-Stage Electricity Demand Modeling Using Machine Learning Algorithms.*Department of Informatics, Faculty of Applied Informatics and Mathematics, Warsaw University of Life Sciences, Nowoursynowska 159, 02-787 Warsaw, Poland.* Energies 2017, 10(10), 1547.

[7] Jui-Sheng Choua, Ngoc-Son Truonga. Cloud forecasting system for monitoring and alerting of energy use by home appliances. *Applied Energy Volume 249, 1 September 2019, Pages 166-17.* https://www.sciencedirect.com/science/article/abs/pii/S0306261919307 10X?via=ihub

[8] S. Singh, A. Yassine and R. Benlamri. "Towards Hybrid Energy Consumption Prediction in Smart Grids with Machine Learning." *2018 4th International Conference*

on Big Data Innovations and Applications (Innovate-Data), Barcelona, 2018, pp. 44-50.

[9] S. Rinchen, A. Yassine, K. Schwartzentruber, H. Ahmed and A. Armitage, ”Integrating Small Scale Green Energy into Smart Grids: Prediction for Peak Load Reduction,” *2018 International Conference on Computer and Applications (ICCA)*, Beirut, 2018, pp. 104-109.

[10] Fahad, M.U.; Arbab, N. Factor Affecting Short Term Load Forecasting. *J. Clean Energy Technol.* 2014, 2, 305-309.

[11] A. Tittaferrante and A. Yassine, ”Multi-Advisor Reinforcement Learning for Multi-Agent Multi-Objective Smart Home Energy Control,” in IEEE Transactions on Artificial Intelligence, doi: 10.1109/TAI.2021.3125918.

[12] N. Sadeghianpourhamami, J. Deleu and C. Develder, ”Definition and Evaluation of Model-Free Coordination of Electrical Vehicle Charging With Reinforcement Learning,” in IEEE Transactions on Smart Grid, vol. 11, no. 1, pp. 203-214, Jan. 2020, doi: 10.1109/TSG.2019.2920320.

[13] F. L. D. Silva, C. E. H. Nishida, D. M. Roijers and A. H. R. Costa, ”Coordination of Electric Vehicle Charging Through Multiagent Reinforcement Learning,” in IEEE Transactions on Smart Grid, vol. 11, no. 3, pp. 2347-2356, May 2020, doi: 10.1109/TSG.2019.2952331.

[14] T. Qian, C. Shao, X. Wang and M. Shahidehpour, ”Deep Reinforcement Learning for EV Charging Navigation by Coordinating Smart Grid and Intelligent Transportation System,” in IEEE Transactions on Smart Grid, vol. 11, no. 2, pp. 1714-1723, March 2020, doi: 10.1109/TSG.2019.2942593.

[15] JR. Vazquez-Canteli JR, Z. Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. Applied Energy 2018;1072â89

[16] S. Vandael, B. Claessens, D. Ernst, T. Holvoet, and G. Deconinck,”Reinforcement learning of heuristic ev fleet charging in a day-ahead electricity market,” IEEE Transactions on Smart Grid, vol. 6, no. 4, pp.1795-1805, July 2015.

[17] F. Tuchnitz, N. Ebell, J. Schlund, M. Pruckner. Development and Evaluation of a Smart Charging Strategy for an Electric Vehicle Fleet Based on Reinforcement Learning. Applied Energy Volume 285, 1 March 2021, 116382

[18] I. Grondman, L. Busoniu, G. A. D. Lopes and R. Babuska, ”A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients,” in IEEE Transactions on Systems, Man, and Cybernetics, Part C

(Applications and Reviews), vol. 42, no. 6, pp. 1291-1307, Nov. 2012, doi: 10.1109/TSMCC.2012.2218595.

[19] S. Singh, A. Yassine and R. Benlamri, "Consumer Segmentation: Improving Energy Demand Management through Households Socio-Analytics," *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech),* Fukuoka, Japan, 2019, pp. 1038-1045.

[20] McLaren, Joyce A, Gagnon, Pieter J, and Mullendore, Seth. Identifying Potential Markets for Behind-the-Meter Battery Energy Storage: A Survey of U.S. Demand Charges. United States: N. p., 2017. Web.

[21] Stephen Haben, Colin Singleton, and Peter Grindrod. Analysis and Clustering of Residential Customers Energy Behavioral Demand Using Smart Meter Data. *IEEE TRANSACTIONS ON SMART GRID, VOL. 7, NO. 1, JANUARY 2016* https://ieeexplore.ieee.org/abstract/document/7063233.

[22] K. Tokuda, S. Matsumoto and M. Nakamura, "Implementing a mobile application for spontaneous peak shaving of home electricity," 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2013, pp. 273-278, doi: 10.1109/WiMOB.2013.6673372.

[23] S. Park and W. -K. Park, "CES peak demand shaving with energy storage system," 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017, pp. 1124-1126, doi: 10.1109/ICTC.2017.8190874.

[24] X. Gong, E. Castillo-Guerra, J. L. Cardenas-Barrera, B. Cao, S. A. Saleh and L. Chang, "Robust Hierarchical Control Mechanism for Aggregated Thermostatically Controlled Loads," in IEEE Transactions on Smart Grid, vol. 12, no. 1, pp. 453-467, Jan. 2021, doi: 10.1109/TSG.2020.3009989.

[25] H. Mortaji, S. H. Ow, M. Moghavvemi and H. A. F. Almurib, "Load Shedding and Smart-Direct Load Control Using Internet of Things in Smart Grid Demand Response Management," in IEEE Transactions on Industry Applications, vol. 53, no. 6, pp. 5155-5163, Nov.-Dec. 2017, doi: 10.1109/TIA.2017.2740832.

[26] K. Mongird, V. Viswanathan, J. Alam, C. Vartanian, V. Sprenkle, R. "2020 grid energy storage technology cost and performance assessment", Publication No. DOE/PA-0204, Dec. 2020

[27] C. Liu, K. T. Chau, D. Wu and S. Gao, "Opportunities and Challenges of Vehicle-to-Home, Vehicle-to-Vehicle, and Vehicle-to-Grid Technologies," in

Proceedings of the IEEE, vol. 101, no. 11, pp. 2409-2427, Nov. 2013, doi: 10.1109/JPROC.2013.2271951.

[28] J. Yusuf, A. S. M. J. Hasan and S. Ula, ”Impacts Analysis & Field Implementation of Plug-in Electric Vehicles Participation in Demand Response and Critical Peak Pricing for Commercial Buildings,” 2021 IEEE Texas Power and Energy Conference (TPEC), 2021, pp. 1-6

[29] Chenxiao Guan, Y. Wang, Xue Lin, S. Nazarian and M. Pedram, ”Reinforcement learning-based control of residential energy storage systems for electric bill minimization,” 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), 2015, pp. 637-642, doi: 10.1109/CCNC.2015.7158054.

[30] H. Xu, W. Kuang, J. Lu and Q. Hu, ”A Modified Incentive-based Demand Response Model using Deep Reinforcement Learning,” 2020 12th IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), 2020, pp. 1-5, doi: 10.1109/APPEEC48164.2020.9220364.

[31] O’Neill, Daniel, et al. ”Residential demand response using reinforcement learning.” 2010 First IEEE international conference on smart grid communications. IEEE, 2010.

[32] Silva, V.A.; Aoki, A.R.; Lambert-Torres, G. Optimal Day-Ahead Scheduling of Microgrids with Battery Energy Storage System. Energies 2020, 13, 5188. https://doi.org/10.3390/en13195188

[33] E. Foruzan, L. Soh and S. Asgarpoor, ”Reinforcement Learning Approach for Optimal Distributed Energy Management in a Microgrid,” in IEEE Transactions on Power Systems, vol. 33, no. 5, pp. 5749-5758, Sept. 2018, doi: 10.1109/TPWRS.2018.2823641.

[34] X. Xu, Y. Jia, Y. Xu, Z. Xu, S. Chai and C. S. Lai, ”A Multi-Agent Reinforcement Learning-Based Data-Driven Method for Home Energy Management,” in IEEE Transactions on Smart Grid, vol. 11, no. 4, pp. 3201-3211, July 2020, doi: 10.1109/TSG.2020.2971427.

[35] S. S. Reka, P. Venugopal, H. H. Alhelou, P. Siano and M. E. H. Golshan, ”Real Time Demand Response Modeling for Residential Consumers in Smart Grid Considering Renewable Energy With Deep Learning Approach,” in IEEE Access, vol. 9, pp. 56551-56562, 2021, doi: 10.1109/ACCESS.2021.3071993.

[36] A. Joshi, H. Kebriaei, V. Mariani and L. Glielmo, "Decentralized Control of Residential Energy Storage System for Community Peak Shaving: A Constrained Aggregative Game," 2021 IEEE Madrid PowerTech, 2021, pp. 1-6, doi: 10.1109/PowerTech46648.2021.9495052.

[37] D. Kaur, R. Kumar, N. Kumar and M. Guizani, "Smart Grid Energy Management Using RNN-LSTM: A Deep Learning-Based Approach," 2019 IEEE Global Communications Conference (GLOBECOM), 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013850.

[38] Li, Y., Gao, W., &amp; Ruan, Y. (2018, April 26). Performance investigation of grid-connected residential PV-battery system focusing on enhancing self-consumption and peak shaving in Kyushu, Japan. Renewable Energy. Retrieved December 18, 2021, from https://www.sciencedirect.com/science/article/pii/S0960148118304920

[39] Pimm, A. J., Cockerill, T. T., &amp; Taylor, P. G. (2018, February 9). The potential for peak shaving on low voltage distribution networks using electricity storage. Journal of Energy Storage. Retrieved December 18, 2021, from https://www.sciencedirect.com/science/article/pii/S2352152X17305601

[40] K. Kaur, A. Dua, A. Jindal, N. Kumar, M. Singh and A. Vinel, "A Novel Resource Reservation Scheme for Mobile PHEVs in V2G Environment Using Game Theoretical Approach," in IEEE Transactions on Vehicular Technology, vol. 64, no. 12, pp. 5653-5666, Dec. 2015, doi: 10.1109/TVT.2015.2482462.

[41] Reddy, T. A., Norford, L. K., &amp; Kempton, W. (2003, August 11). Shaving residential air-conditioner electricity peaks by intelligent use of the building thermal mass. Energy. Retrieved December 18, 2021, from https://www.sciencedirect.com/science/article/abs/pii/036054429190060Y

[42] J. Dong, A. Yassine and A. Armitage, "Image-based with Peak Load Ensemble Prediction System for Demand Response in Smart Grid," 2021 International Symposium on Networks, Computers and Communications (ISNCC), 2021, pp. 1-6, doi: 10.1109/ISNCC52172.2021.9615837.

[43] J. Dong, A. Yassine and A. Armitage, "Image-Based Processing Mechanism for Peak Load Forecasting in Smart Grids," 2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE), 2020, pp. 64-69,

[44] T. Hong, P. Pinson, and S. Fan, "Global energy forecasting competition 2012," Int. J. Forecast., vol. 30, no. 2, pp. 357-363, 2014.

[45] A. Goia, C. May, and G. Fusai, ”Functional clustering and linear regression for peak load forecasting,” Int. J. Forecast., vol. 26, no. 4, pp. 700-711, 2010.

[46] D. C. Montgomery, C. L. Jennings, and M. Kulahci, Introduction to Time Series Analysis and Forecasting. Somerset, US: Wiley, 2011.

[47] R. Ngabesong and L. McLauchlan, ”Implementing ”R” Programming for Time Series Analysis and Forecasting of Electricity Demand for Texas, USA,” 2019 IEEE Green Technologies Conference(GreenTech), 2019, pp. 1-4, doi: 10.1109/GreenTech.2019.8767131.

[48] H. Huang, Y. Cai, H. Xu and H. Yu, ”A Multiagent Minority-Game-Based Demand-Response Management of Smart Buildings Toward Peak Load Reduction,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 4, pp. 573-585, April 2017.

[49] L. Barbierato et al., ”A Distributed IoT Infrastructure to Test and Deploy Real-Time Demand Response in Smart Grids,” in IEEE Internet of Things Journal, vol. 6, no. 1, pp. 1136-1146, Feb. 2019.

[50] A. Yassine, S. Singh and A. Alamri, ”Mining Human Activity Patterns From Smart Home Big Data for Health Care Applications,” *in IEEE Access*, vol. 5, pp. 13131-13141, 2017.

[51] Singh S., Yassine A. Big data mining of energy time series for behavioral analytics and energy consumption forecasting *Energies*, 11 (2018), p. 452

[52] S. Singh and A. Yassine, ”Mining Energy Consumption Behavior Patterns for Households in Smart Grid,” *in IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 3, pp. 404-419, 1 July-Sept. 2019.

[53] K. Simonyan and A. Zisserman, ”Very Deep Convolutional Networks for Large-Scale Image Recognition.” Retrieved from https://arxiv.org/abs/1409.1556 ,2015, April 10.

[54] Zhang, W., Liu, H., Wang, F., Xu, T., Xin, H., Dou, D., & Xiong, H. (2021). Intelligent electric vehicle charging recommendation based on multi-agent reinforcement learning. In The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021 (pp. 1856-1867). (The Web Conference 2021 - Proceedings of the World Wide Web Conference, WWW 2021). Association for Computing Machinery, Inc. https://doi.org/10.1145/3442381.3449934

[55] Kaiqing Zhang, Zhuoran Yang, Tamer Basar. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms, arXiv:1911.10635.

[56] Sutton, Richard S., and Andrew G. Barto. Introduction to reinforcement learning. Vol. 135. Cambridge: MIT press, 1998.

[57] Sutton, Richard S.. "Learning to Predict by the Methods of Temporal Differences." Machine Learning 3 (2005): 9-44.

[58] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8(3-4):279?292, 1992.

[59] Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM,38(3):58?68, 1995.

[60] Tsitsiklis J N, Van R B. An analysis of temporal-difference learning with function approximation. IEEE Transactions on Automatic Control, 1997, 42(5): 674-690

[61] Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. Journal of Machine Learning Research, 5:1063?1088, 2004.

[62] Nicolas Heess, David Silver, and Yee Whye Teh. Actor-critic reinforcement learning with energy-based policies. In European Workshop on Reinforcement Learning, page 43, 2012.

[63] Riedmiller M. Neural fitted q iteration-first experiences with a data efficient neural reinforcement learning method. Proceedings of the Conference on Machine Learning. Berlin, German, 2005: 317-328

[64] Sascha Lange and Martin Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In Neural Networks (IJCNN), The 2010 International Joint Conference on, pages 1?8. IEEE, 2010.

[65] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

[66] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou. Playing Atari with Deep Reinforcement Learning. NIPS 2013.

[67] Mnih, Volodymyr, et al. Human-level control through deep reinforcement learning. Nature. 518 (7540): 529-533, 2015.

[68] Van H V, Guez A, Silver D. Deep reinforcement learning with double q-learning. Proceedings of the AAAI Conference on Artificial Intelligence. Phoenix, USA, 2016: 2094-2100.

[69] Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. Proceedings of the 4th International Conference on Learning Representations. San Juan, Puerto Rico, 2016:322-355.

[70] Wang Z, Freitas N D, Lanctot M. Dueling network architectures for deep reinforcement learning. Proceedings of the International Conference on Machine Learning. New York, USA, 2016: 1995-2003.

[71] Hausknecht M, Stone P. Deep recurrent q-learning for partially observable MDPs. arXiv preprint arXiv:1507.06527, 2015.

[72] A. Abdulaal, M. H. Cintuglu, S. Asfour and O. A. Mohammed, "Solving the Multivariant EV Routing Problem Incorporating V2G and G2V Options," in IEEE Transactions on Transportation Electrification, vol. 3, no. 1, pp. 238-248, March 2017, doi: 10.1109/TTE.2016.2614385.

[73] H. Yang, Y. Deng, J. Qiu, M. Li, M. Lai and Z. Y. Dong, "Electric Vehicle Route Selection and Charging Navigation Strategy Based on Crowd Sensing," in IEEE Transactions on Industrial Informatics, vol. 13, no. 5, pp. 2214-2226, Oct. 2017, doi: 10.1109/TII.2017.2682960.

[74] Egorov, Maxim. "Multi-agent deep reinforcement learning." CS231n: convolutional neural networks for visual recognition (2016): 1-8.

[75] Solan, Eilon, and Nicolas Vieille. "Stochastic games." Proceedings of the National Academy of Sciences of the United States of America vol. 112,45 (2015): 13743-6. doi:10.1073/pnas.1513508112

[76] Busoniu L., Babuska R., De Schutter B. (2010) Multi-agent Reinforcement Learning: An Overview. In: Srinivasan D., Jain L.C. (eds) Innovations in Multi-Agent Systems and Applications - 1. Studies in Computational Intelligence, vol 310. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14435-67

[77] Wang, Ying, and Clarence W. De Silva. "Multi-robot box-pushing: Single-agent q-learning vs. team q-learning." 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006.

[78] H. Zhang, H. Jiang, Y. Luo and G. Xiao, "Data-Driven Optimal Consensus Control for Discrete-Time Multi-Agent Systems With Unknown Dynamics Using Reinforcement Learning Method," in IEEE Transactions on Industrial Electronics, vol. 64, no. 5, pp. 4091-4100, May 2017, doi: 10.1109/TIE.2016.2542134.

[79] Galindo-Serrano, Ana, and Lorenza Giupponi. "Distributed Q-learning for aggregated interference control in cognitive radio networks." IEEE Transactions on Vehicular Technology 59.4 (2010): 1823-1834.

[80] Littman, Michael L. "Markov games as a framework for multi-agent reinforcement learning." Machine learning proceedings 1994. Morgan Kaufmann, 1994. 157-163.

[81] Hu, Junling, and Michael P. Wellman. "Nash Q-learning for general-sum stochastic games." Journal of machine learning research 4.Nov (2003): 1039-1069.

[82] Greenwald, Amy, Keith Hall, and Roberto Serrano. "Correlated Q-learning." ICML. Vol. 3. 2003.

[83] Littman, Michael L. "Friend-or-foe Q-learning in general-sum games." ICML. Vol. 1. 2001.

[84] Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in neural information processing systems 30 (2017).

[85] Wang, Y. Witten, I. H. (1996). Induction of model trees for predicting continuous classes. (Working paper 96/23). Hamilton, New Zealand: University of Waikato, Department of Computer Science.

[86] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. 2016.

[87] Kuhn, Max, et al. "Cubist models for regression." R package Vignette R package version 0.0 18 (2012): 480.

[88] Svozil, Daniel, Vladimir Kvasnicka, and Jiri Pospichal. "Introduction to multi-layer feed-forward neural networks." Chemometrics and intelligent laboratory systems 39.1 (1997): 43-62.

[89] Hochreiter, Sepp, and JÃ¼rgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.