Artificial Intelligence Based Smart Communities with Privacy Enhancement

WENZHI CHEN

Artificial Intelligence Based Smart Communities with Privacy Enhancement

Wenzhi Chen

A Thesis presented for the degree of
Doctor of Philosophy

Department of Engineering
Durham University
United Kingdom
September 2024

Abstract

This thesis introduces Artificial intelligence-based smart communities, including the optimization of smart home and home microgrid applications, thus allowing communities to be more comfortable and affordable for their residents while protecting user privacy. For these purposes, a three-layer smart community framework is proposed.

The smart home layer focuses on a household device action recommendation system. The household appliance data is organized into a knowledge graph. A framework, 'DARK' (Device Action Recommendation with Knowledge graph), is proposed, including three parts. Firstly, a household device action recommendation algorithm is proposed to make accurate household appliance recommendations. Next, graph interpretable characteristics are developed in DARK using trained graph embeddings. Lastly, with the recommendation expectations, the consumers' degree of satisfaction and the appliances' average power load are modelled as a multi-objective optimization problem in DARK to participate in demand response.

The home microgrid layer concentrates on the home microgrid scheduling algorithm. The proposed microgrid model includes energy storage systems, PV panels, loads, and the connection to the main grid. Firstly, a multi-objective deep reinforcement learning architecture is proposed for accumulated carbon emissions and electricity costs optimization. Secondly, data privacy is protected by federated learning, in which the original data is not uploaded to the server but remains locally stored. Finally, the optimization results are selected and stored to form Pareto fronts, allowing users to bias towards their preferred optimization goals.

The privacy protection layer focuses on the measurement of potential privacy leakage in federated learning. With the usage of an explainable algorithm, a framework named SAFE-Home (Smart Applications in Federated learning with Emphasis on Home privacy) is proposed to locate the key factors that affect privacy leakage and vulnerable data in federated smart communities. The theoretical analysis and simulation results are consistent, demonstrating the similarity in the privacy leakage trends in different applications and situations.

Declaration

The work in this thesis is based on research carried out at the Department of Engineering, Durham University, United Kingdom. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Parts of this work have been published in the form of refereed papers:

Chapter 3

Conference paper:

Chen, Wenzhi, Hongjian Sun, Minglei You, and Jing Jiang. "Accurate Action Recommendations and Demand Response for Smart Homes via Knowledge Graphs," in *Proceedings of the 2024 IEEE International Conference on Industrial Technology (ICIT 2024)*, vol.2, pp.1-6, Bristol, UK, March 2024.

Journal paper:

Chen, Wenzhi, Hongjian Sun, Minglei You, Jing Jiang and Marco Rivera. "DARK: Device Action Recommendation System with Knowledge Graph", *the paper is in the pre-submission stage.*

Chapter 4

Conference paper:

Chen, Wenzhi, Hongjian Sun, Jing Jiang, Minglei You, and P. William JS. "Protecting privacy in microgrids using federated learning and deep reinforcement learn-

ing," in *Proceedings of the 2022 IET International Conference on Advances in Power System Control, Operation and Management (APSCOM 2022) conference*, vol.1, pp.205-210, Hong Kong, 2022.

Chapter 5

Conference paper:

Chen, Wenzhi, Hongjian Sun, Minglei You, and Jing Jiang. "Privacy Leakage in Federated Home Applications Using Gradient Inversion Algorithms," in *Proceedings of the 2024 IEEE International Conference on Industrial Technology (ICIT 2024)*, vol.3, pp.1-6, Bristol, UK, March 2024.

Journal paper:

Chen, Wenzhi, Hongjian Sun, Minglei You, Jing Jiang and Marco Rivera. "SAFE-Home: Smart Applications Federated Learning with Emphasis on Home Privacy Against Gradient Inversion Algorithms", *the paper is in the pre-submission stage.*

Acknowledgements

I am especially grateful to Professor Hongjian Sun, whose support and guidance has allowed me to conduct my research in Durham University. Professor Sun is not only knowledgeable and patient, but he has also provided me with an open research environment, while respecting all my ideas. Whenever I have encountered any difficulties, he has always offered me valuable advice and encouragement. His outlook on life and his values have had a great influence on me, laying a solid foundation for my future career. I regard him as both my supervisor as well as my mentor. I feel immensely privileged to have been his student.

Next, I would like to express my gratitude to Minglei You. Not only are we both alumni of the Beijing University of Posts and Telecommunications and Durham University, but we also share many common interests. He is the one who has supported me during my PhD research and academic life, and I cherish this mentor-like friendship. Whether it's daily life issues or academic challenges, he has always immediately lent a helping hand to me. His academic rigor, his open-minded approach to life, and passionate enthusiasm have deeply influenced me.

Thirdly, I want to show my deep appreciation to both Professor Jing Jiang and Professor Marco Rivera for their meticulous guidance on my conference presentations and journal extensions, ensuring that my research results were accurately expressed and successfully published.

Additionally, I extend my heartfelt thanks to my lab mates and friends in Durham, including Weiqi Hua, Yingjia Huang, Weizhen Li, Xinzhuo Zhang, Sunny Du and Ken, Pina-Gongora Diana, Aguilar-Celis Alexis, Harsh Pratik, Jinjie Liu, Rong Lin, Chenguang Liu (in the order we met.), for their care and support.

I want to thank my family in China, including my grandparents and my parents, as well as in-laws, for their continuing love and assistance over many years. Especially, I want to thank my dear wife, Ruobing Liu, in Beijing. I am truly grateful to her for her understanding and loving support during my PhD studies in the UK.

Finally, I would like to thank Durham University. Special thanks to St. John's College and the Department of Engineering for their encouragement and help throughout my academic journey.

Contents

List of Figures

List of Tables

List of Symbols

$\mathbb{A}$	The action space for the DQN algorithm.
A	The effective surface area of the PV panel.
$\mathbf{Adj}$	The adjacency matrices in knowledge graph algorithm.
$\mathbf{Adj}_h$	The adjacency matrices in knowledge graph algorithm from the head embedding.
$\mathbf{Adj}_t$	The adjacency matrices in knowledge graph algorithm from the tail embedding.
a	The a-th layer inside the neural networks.
act	The action in DQN.
adt	The optimization rounds for the ADAM optimizer in the GS algorithm.
agg	The aggregated embedding in KGAT.
agg_h	The aggregated embedding from head embedding in Algorithm 1.
agg_t	The aggregated embedding from tail embedding in Algorithm 1.
ai	The activate value of the input gate in LSTM.
$atn_{(h,r,t)}$	The attention to knowledge graph algorithm aggregation.
$atn_{h\text{-}(h,r,t)}$	The attention calculated from the head embeddings in knowledge graph in Algorithm 1.

$atn_{t\text{-}(h,r,t)}$	The attention calculated from the tail embeddings in knowledge graph in Algorithm 1.
$\mathbb{B}_{\text{out}}$	The output of the Bellman equation.
B_i	The biases for DNN in the KGAT algorithm.
b	The bias of neural networks.
b_a	The bias of the a-th fully connected layer for the DNN.
b_{fg}	The bias for the forgotten gate of the LSTM.
b_{gc}	The bias for the cell candidate of the LSTM.
b_{in}	The bias for the input gate of the LSTM.
b_o	The bias for the output gate of the LSTM.
b_U	The bias for the fully connected layer of the LSTM.
bch	The batch size in GS attack, also one of the inputs in explainable layer I in Algorithm 3.
$\mathbb{C}_1$	The capacity to store memory replay in Algorithm 2.
$\mathbb{C}_2$	The capacity to store possible Pareto fronts in Algorithm 2.
Ca	The carbon emission data from the dataset.
C_B	The maximum battery capacity.
CE	The accumulated carbon emissions in the DQN training steps.
$\mathrm{c}(\cdot)$	The combination calculations in mathematics.
cs	The cell state of the LSTM.
$\mathbb{D}$	The randomly initialized data (dummy data) for the GS attack in federated learning.
$\mathrm{DQN}_{\text{online}}(\cdot)$	The online network of the DQN.
$\mathrm{DQN}_{\text{target}}(\cdot)$	The target network of the DQN.
DtV	The data volume of all clients in federated learning.
d	The dimension.
d_a	The dimension of the neural network layers a.
d_E	The dimension of embedding in the KGAT algorithm.
d_{in}	The input dimension of the layer of DNN in DQN.

d_{out} The output dimension of the layer of DNN in DQN.

d_R The dimension of the relation in the KGAT algorithm.

df_i The dissatisfaction factor in energy and user's expected satisfaction optimization in Chapter 3.

E_B The battery energy level.

E_{bought} The amount of energy bought from the main power grid to charge the battery.

E_{ch} The energy used to charge the battery.

E_{dis} The discharging energy used by the load.

E_L The energy used by the active load.

E_{MG} The total energy purchased from the main power grid.

E_{max} The maximum limit of energy to charge a battery within a time interval.

E_{maxd} The maximum limit of energy to discharge a battery within a time interval.

E_{PV} The energy generated from PV.

EC The accumulated electricity cost in DQN training steps.

e Natural logarithm.

em_h The embedding of heads in the knowledge graph algorithm.

$em_{h\text{-}Nh}$ The embedding in the Algorithm 1 with attention weighted from the head embedding part.

$em^{se}_{(h,r,t,\text{time})}$ The embedding of different sequences in the knowledge graph algorithm.

em^{se}_{time} The time embedding, representing the time that the appliance in agg_h and agg_t were using.

$em^{ti}_{(h,r,t)}$ The embedding of different time zones in the knowledge graph algorithm.

$em^{lo}_{(h,r,t)}$ The embedding of different locations in the knowledge graph algorithm.

em_{Nh} The embedding of the aggregated entity in the knowledge graph algorithm.

em_r The embedding of relations in the knowledge graph algorithm.

em_t The embedding of tails in the knowledge graph algorithm.

$em_{t\text{-}Nh}$ The embedding in Algorithm 1 with attention weighted from tail embedding part.

em^{se}_{total} The product value of the embeddings for the reason 'sequence' as the explanation in the household device action recommendation system.

$em^{se}_{(h,r,t)}$ The head, relation tail embeddings for the reason 'sequence' in the household device action recommendation system.

em^{ti}_{total} The product value of the embeddings for the reason 'time' as the explanation in the household device action recommendation system.

$em^{ti}_{(h,r,t)}$ The head, relation tail embeddings for the reason 'time' in the household device action recommendation system.

em^{lo}_{total} The product value of the embeddings for the reason 'location' as the explanation in the household device action recommendation system.

$em^{lo}_{(h,r,t)}$ The head, relation tail embeddings for the reason 'location' in the household device action recommendation system.

F_{carbon} The function represents the value of the carbon emissions.

F_{cost} The function represents the value of the electricity costs.

F_{pareto} The function of the Pareto front.

Flr The load and voltage forecasting or household devices action recommendation model in federated learning.

f_{avgpower} Averaged power function.

$f_{\text{satisfaction}}$ Satisfactory function.

$f_{\text{dissatisfaction}}$ Dissatisfaction function.

f_{im} The model of explainable layer I represents an interpretable model.

f_{load} The load model in Chapter 4.

f_{nim} The model of explainable layer I represents a non-interpretable model.

fg The forgotten gate of the LSTM.

G The size of the knowledge graph.

$g_{(h,r,t)}$ The positive knowledge graph triples.

$g_{(h,r,t_{\mathbb{D}})}$ The negative knowledge graph triples by replacing t with a random value $t_{\mathbb{D}}$.

gc The cell candidate in LSTM represents new information that may be added to the cell state.

h Head (for the knowledge graph).

hs The hidden state of the RNN and LSTM.

hsi The hidden-layer size.

I The number of clients in federated learning in Chapter 5.

i Used for counting, for example, i-th client in Chapter 4.

ig The training iterations in GS attack, also one of the inputs in explainable layer I.

J The objective function of DQN. This is used to minimize the total carbon emissions and electricity costs.

J_{cost} The objective function (reward) of the electricity costs in DQN.

J_{carbon} The objective function (reward) of the carbon emissions in DQN.

j The objective function within a time slot to minimize the total carbon emissions and electricity costs according to rewards.

K The volume of data for updating the memory replay in federated learning.

k_{FL} The length of the federated learning training iterations.

k_n The length of the time interval.

$\mathbb{L}$ The list to store the recommendation results in Algorithm 1.

$\nabla L(\cdot)$ The batch gradient in federated learning training.

$L(\cdot)_{\text{equal}}$ The loss function, for situations where the batch size and training iterations are equal. The data is directly input to LIME as perturbed data without any weighting.

$L(\cdot)_{\text{lime}}$ The loss function for the LIME algorithm.

$L(\cdot)_{\text{proposed}}$ The total loss function for the proposed explainable machine learning algorithm in Chapter 5.

$L(\cdot)_{\text{similiar}}$ The loss function, for situations where batch sizes and training iterations are similar, the weighted data is input to LIME.

$l(\cdot)$ The loss function (for example, mean squared error loss).

length The length of a sequence or the number of statistical objects.

lo	The 'location' reason for interpretation, means two appliances used together because they are in the same location.
$\mathbb{M}_p$	The pareto memory set of DQN.
$\mathbb{M}_r$	The replay memory set of DQN.
M	The repeat turn in the proposed Algorithms.
m	The mini-batched labels in federated learning.
$m_{\mathbb{D}}$	The mini-batched randomly initialized (dummy) labels.
$m_{\mathbb{D}*}$	The mini-batched optimized (recovered) labels, which are closer to the real labels after optimization.
mb	The mini-batched input.
$mb_{\mathbb{D}}$	The mini-batched randomly initialized (dummy) data.
$mb_{\mathbb{D}*}$	The mini-batched optimized (recovered) data, which is closer to the real data after optimization.
N	Used for counting.
$\mathbb{N}$	The set of natural elements.
N_{bch}	The number of batch sizes in Algorithm 3.
N_c	The number of clients in federated learning in Chapter 4.
N_{DQN}	The number of the training iterations in DQN.
N_{exp}	The number of features in the proposed algorithm in Chapter 5.
N_{iter}	The number of training iterations in Algorithm 3.
N_{PC}	The number of all parameters for a client.
N_{PF}	The length of all the data which is possible to be in a Pareto front.
N_{rep}	The number of repeated times in Algorithm 3.
Nh	The aggregated embeddings in KGAT with attention weighted.
n	The time interval.
$\mathbb{O}(\cdot)$	Algorithm time complexity.
O	The output of the recommendation result in Algorithm 1.
o	The output gate of the LSTM.
o_v	The output gate of the LSTM with sequence v.

org	Original data in Algorithm 3, Chapter 5.
org_{bch}	Batched original data in Algorithm 3, Chapter 5.
$\mathbb{P}$	The feasible set of criterion vectors in $\mathbb{R}^M$.
$\mathbb{PA}$	The set of the feasible decisions in the Pareto front.
$P_{\text{charge}}(\tau)$	The charging power rate of batteries in time τ.
$P_{\text{discharge}}(\tau)$	The discharging power rate of batteries in time τ.
P_i	The rated power of appliances.
P_i^{real}	The real power of appliances.
$P_L(\tau)$	The load power in time τ.
$P_{\text{MG}}(\tau)$	The power flow from the main grid in time τ.
$P_{\text{net}}(\tau)$	The net power state of batteries in time τ.
$P_{\text{PV}}(\tau)$	The PV power generated in time τ.
$Pr(n)$	The electricity price data from the dataset in time interval n.
p_1 and p_2	Used for demonstrating the points in the L2 distance, cosine similarity and Pareto front.
pa	The feasible decisions in the Pareto front.
pe	$pe = 1$ if the main power grid is used to power the loads and $pe = 0$ if the batteries are used.
$Q_{\max}(\cdot)$	The maximum Q-value considers all the possible situations.
$\mathbb{R}^M$	The metric space in the Pareto front.
R	Different reasons for the allocation from the embeddings.
R_{lo}	The reason 'location' for the allocation from the embeddings.
R_{se}	The reason 'sequence' for the allocation from the embeddings.
R_{ti}	The reason 'time' for the allocation from the embeddings.
Reward$(\cdot)$	The reward function for the proposed DQN.
r	Relation (for the knowledge graph).
rec	The recovered data in Algorithm 3, Chapter 5.
rec_{bch}	The batched recovered data in Algorithm 3, Chapter 5.
rd	The instant reward given to the DQN.

$\mathbb{S}$	The state space of the DQN.
$S_{\mathbb{A}}$	The action space size.
SoC	The state of charge of the battery.
SoC_{lim}	The limitations of the battery's state of charge.
SP	The space complexity.
SP_{AC}	The space complexity in attention calculations.
SP_{AO}	The space complexity in aggregation operations.
SP_{DNN}	The space complexity for the DNN algorithm.
SP_{KGAT}	The space complexity for the KGAT algorithm.
$SR(n)$	The averaged solar irradiance (W/m^2).
s_h	The sampled dataset from em_h^{se} in Algorithm 1.
s_i	The sampled dataset in Algorithm 2.
s_t	The sampled dataset from em_t^{se} in Algorithm 1.
se	The 'sequence' reason for interpretation, means two appliances always used one after another.
$(\cdot)^T$	The transpose operation.
Tr_{gap}	Updating the online network with the target network regularly every Tr_{gap} rounds.
Tr_{renew}	The target network is trained with the stored memory every Tr_{renew} rounds.
Tr_{upload}	The federated learning performs the FedAvg algorithm and generates the Pareto front every Tr_{upload} rounds.
$TransE$	The embedding trained by the TansE algorithm.
$TransE_h$	The head embedding trained by the TansE algorithm.
$TransE_r$	The relation embedding trained by TansE algorithm.
$TransE_t$	The tail embedding trained by the TansE algorithm.
t	Tail (for the knowledge graph).
$t_{\mathbb{D}}$	Replacing the knowledge graph tail with $t_{\mathbf{D}}$ for negative sampling.
ti	The 'time' reason for interpretation, means two appliances often used at the same time.

U_v The output layer of the LSTM.

u The input layer of the LSTM.

u_v The input sequence of the LSTM.

V_{Ra} The battery voltage.

Vec The input vector of the DNN in the KGAT in algorithm 1.

v The data sequence in LSTM.

W_a The weights of the deep learning algorithm in layer a.

$W_{a,row,col}$ The weights of the convolution kernel in CNN in layer a.

W_{DNN} The DNN is fitted to the Q-table of the DQN, and W_{DNN} is the weights (model) of the DNN.

W_{DQN} The parameters (weights) of the DQN.

W_{fg} The weights for the forgotten gate of the LSTM.

W_{gc} The weights for the cell candidate of the LSTM.

W_{hh} The weights matrix for transitioning from the previous output to the current hidden state in RNN.

W_{ho} The weights from the hidden states to the output in LSTM.

W_{hy} The weights matrix for the transition from the current input to the hidden state in RNN.

W_{in} The weights for the input gate of the LSTM.

W_o The weights for the output gate of the LSTM.

W_r The relation weights matrix in KGAT.

W_{trans} The transformation weights matrix in KGAT.

W_u The weights for the DNN in KGAT.

W_v The output weights matrix of the LSTM.

w The model parameter, specifically refers to weights of the deep learning models.

w_{averaged} The averaged global model parameters in federated learning.

w_a The weights in layer a.

$w_{\mathbb{D}}$ The model parameter in federated learning, including weights and biases trained by the randomly initialized (dummy) data and labels.

w_i The model parameter in federated learning, including weights and biases for client i.

$w(n)$ The model parameter, including weights and biases during the time interval n.

w_{online} The model parameter of the DQN online network.

w_{target} The model parameter of the DQN target network.

x_a The output of a-th fully connected layer for the DNN.

x_a^{bch} The output of a fully connected layer for the DNN with mini-batched data.

x_{final} The output of final fully connected layer for the DNN.

y_a The input of deep learning layer a, as well as the output of deep learning layer $a-1$.

y_a^{bch} The input of deep learning layer a, with mini-batched data.

$y_{a-1,i+row,j+col}$ The elements of the input feature map that the kernel covers in CNN.

y_{final} The input of the final deep learning layer.

$y(\tau)$ The output at time step τ.

z The original input data of explainable layer II.

ze The data in the group situations where batch size and training iterations are equal, in explainable layer II.

zp The dithered input data of explainable layer II.

zp_i^* The set of the dithered input data of explainable layer II.

zs The data in the group where batch size and training iterations are similar, in explainable layer II.

Φ The activation function.

$\Phi_a(\cdot)$ The activation function for the DNN.

$\Phi_{\text{final}}(\cdot)$ The final-layer activation function for the DNN.

α The bias towards instant rewards or future rewards in DQN.

β For the Kendall coefficient, the number of orderly rankings of the recovered 1. load and voltage data or 2. the embeddings is β.

γ The discount factor of DQN.

ϵ The possibility of choosing a random action in the ϵ-greedy algorithm of DQN.

η_b The battery efficiency.

η_c The inverter efficiency of the battery.

η_{FL} The learning rate of the federated learning.

η_{lr} The learning rate.

η_{PV} The solar-electric energy conversion efficiency of the PV panel.

η_s The charging efficiency of the battery.

λ The bias towards optimizing accumulated carbon emissions or electricity costs.

μ_L The mean value of the active power.

π The policy, represents a mapping between the state and the action space.

σ_L The standard deviation of the active power.

τ Representing the continuous time.

ϕ The preprocessed dataset in algorithm 2.

ψ For Kendall coefficient, the numbers of disorderly rankings of 1. the recovered load and voltage data or 2.embeddings is ψ.

ζ The hyperparameter in LIME, and a larger value of ζ will rapidly decrease the weights of samples farther away.

$(\cdot)'$ The differentiation operation.

$||\cdot||_2^2$ The squared results of the L2 distance.

List of Abbreviations

AI	Artificial intelligence
CAM	Class activation mapping
CNN	Convolutional neural network
DARK	Proposed framework in Chapter 3, device action recommendation system with knowledge graph
DDPG	Deep deterministic policy gradient
DLG	Deep leakage from gradients
DNN	Deep neural network
DQN	Deep-Q network
DR	Demand response
EU	The European Union
FedAvg	Federated averaging algorithm
GAN	Generative adversarial network
GNN	Graph neural network
Grad-CAM	Gradient-weighted class activation mapping
GraphSAGE	Graph sample and aggregate
GS	Gradient similarity

HPTC	High performance technical computing, the projector in home.
iDLG	Improved deep leakage from gradients
KGAT	Knowledge graph attention network
KGCN	Knowledge graph convolutional network
L-BFGS	Limited memory Broyden Fletcher Goldfarb Shanno
LIME	Local interpretable model-agnostic explanations
LRP	Layer-wise relevance propagation
LSTM	Long short-term memory
MA-DDPG	Multi-agent deep deterministic policy gradient
PID	Proportional integral derivative control
PV	Photovoltaic
RNN	Recurrent neural network
SAFE-Home	Proposed framework in Chapter 5, smart applications federated learning with emphasis on home privacy
SHAP	Shapley additive explanations
SoC	State of charge
TransE	Translating embeddings for modelling multi-relational data
UK	The United Kingdom
UK-DALE	UK domestic appliance-level electricity

Dedicated to my family, 1401 King Tiger

including our stuffed animals,

and my dear wife, Ruobing.

CHAPTER 1

Introduction

This thesis presents an Artificial Intelligence (AI)- based smart community system. It introduces the general background, followed by the research objectives, and then the research contributions. Finally, the outline of the thesis structure is listed.

1.1 General backgrounds

This thesis focuses on the study of smart communities. The concept of a smart community consists of two parts: the smart home system within individual households and the home microgrid formed from multiple households. The overall goal of a smart community is to enhance the residents' quality of life through the application of smart home systems while promoting energy efficiency and the reduction of emissions via the home microgrid.

The term 'smart home' refers to a home automation system that utilizes advanced technologies, integrates individual needs, and achieves intelligent control. According to Statista, the global smart home market is expected to generate $154.4 billion in revenue by end of 2024, making it a very promising research direction [1]. Research in the field of smart homes can significantly improve the quality of life

of residents. For instance, smart homes can enhance home safety through real-time environmental monitoring, such as automatically alerting the residents when a gas leak is detected to prevent potential danger. Another research area is that they can also provide accessible living support, such as by automating household appliances to offer more convenient lifestyles for family members with mobility difficulties. Additionally, smart homes can reduce power consumption through demand-side algorithms [2]. Overall, smart home technology can create a more satisfying, energy-efficient, and personalized living space.

Research on the home microgrid can promote energy conservation and emissions reduction, which have received considerable attention by many countries. For example, the European Union Emissions Trading Scheme has committed these countries to becoming carbon-neutral by 2050 [3]. The home microgrid is an excellent solution in this regard. A home microgrid refers to a small-scale microgrid for several households, consisting of distributed energy, individual household loads, storage devices, and control devices. Unlike traditional power grids, microgrids utilize renewable energy sources, such as solar and wind energy, instead of only relying on traditional grid energy sources like oil, coal, and natural gas. Furthermore, energy scheduling can be achieved through an energy management system, such as peak-shaving or purchasing electricity when prices are low, via scheduling algorithms [4] [5]. Overall, it is necessary to develop microgrid algorithms that increase renewable energy penetration while at the same time optimizing performance, further reducing cumulative electricity costs and the production of carbon emissions.

Moreover, since the data used in smart communities comes from the residents, it is inherently private. For individuals, data leakage could lead to identity theft, financial loss, and even personal security threats. To protect data privacy, the European Union has introduced the General Data Protection Regulation and the United Kingdom has enacted the Data Protection Act [6] [7]. Federated learning is the preferred method for protecting data privacy in AI systems; however, as privacy technologies evolve, studies have found that federated learning still poses risks [8]. It is necessary to address any potential attacks that could lead to privacy breaches and to measure the extent of any possible privacy leakage. Therefore, data privacy

protection and the measurement of privacy leakage are another focus of this thesis.

Artificial intelligence is a crucial tool for achieving all of the aforementioned goals within the smart community due to its powerful data analytical and optimization capabilities. Specifically, deep learning algorithms in artificial intelligence are well-suited to prediction and classification tasks, such as electricity load forecasting. Reinforcement learning algorithms are suitable for control tasks, such as scheduling algorithms in microgrids. Optimization algorithms are adept at finding optimal solutions to complex problems, such as balancing conflicting objectives in power systems [9] [10]. Many application scenarios in power systems can be perfectly integrated with artificial intelligence algorithms. Therefore, this thesis focuses on combining these artificial intelligence algorithms with each part of the smart community including smart homes, home microgrids and privacy preservation issues.

1.2 Research objectives

Based on the backgrounds above, in this thesis, the research firstly focuses on the home area—household device action recommendation systems. Then, it expands from the home area to the home microgrid area and focuses on microgrid scheduling. Finally, the factors affecting privacy leakage in the context of household-distributed machine learning are researched based on the first two research directions. Specifically, the focus is placed on the following three research objectives.

1.2.1 Household device action recommendation system

The device action recommendation system is part of the research scope of household appliance automation and smart homes, which are dedicated to making daily life more convenient and electricity usage more efficient. This research aims:

1. To develop a recommendation system for predicting the next possible household device action accurately.

2. To develop an interpretation method for the recommended results, making the proposed recommendation system more reliable.

3. To develop demand response optimization based on the recommended results, enhancing the household energy efficiency.

1.2.2 Home microgrid scheduling systems

Home microgrid scheduling systems are part of the research in the area of microgrid optimization, aimed at enhancing energy efficiency, reducing cumulative electricity costs, while lowering carbon emissions. This research aims:

1. To develop a smart energy scheduling algorithm, while considering energy costs and carbon emissions optimization.

2. To develop a distributed algorithm for collaborations between smart homes, while protecting the privacy of home data.

3. To develop a method to balance different optimization objectives.

1.2.3 Privacy protection

Home privacy protection belongs to the cross-research area covering both cybersecurity and smart homes (as well as home microgrids). Although privacy protection algorithms already exist in 1.2.2, this part assumes the presence of malicious attackers who can further compromise the privacy. Under this assumption, this part is aimed at measuring the privacy leakage of both home data and microgrid data under federated learning attacks, as well as giving residents suggestions to protect their privacy. This research aims:

1. To develop a framework for measuring and predicting privacy leakage in federated home applications.

2. To make theoretical analysis of the Gradient Similarity (GS) attack in federated learning.

3. To give residents advice for privacy preservation.

To achieve the goals above, research has been carried out for each objective, and an overall architecture has been designed, which will be introduced in the next subsection.

1.3 Research contributions

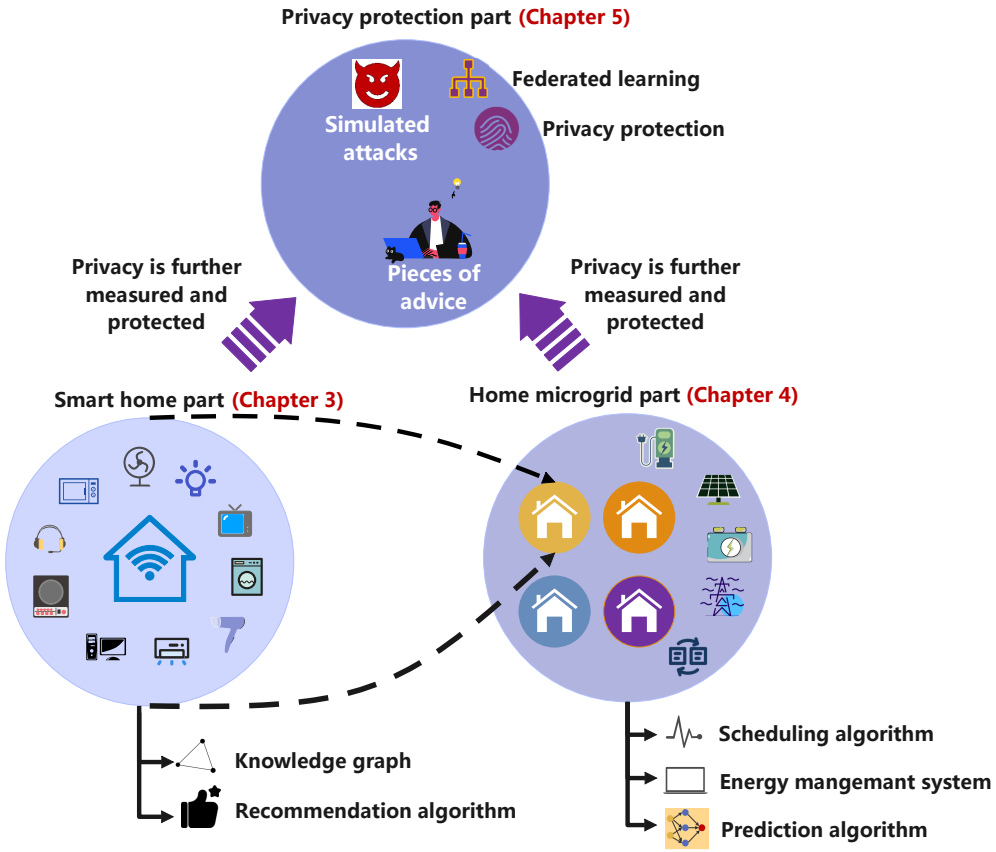

Figure 1.1: The proposed architecture, including the smart home part, home microgrid part and privacy protection part.

The work in this thesis can be summarized as a three-part model: the smart home part, the home microgrid part, and the privacy protection part. As shown in Fig. 1.1, the bottom left corner is the smart home part, responsible for performing the household device action recommendations and demand response optimizations. The bottom right corner shows the home microgrid part, which collects data from the microgrid and performs bi-objective optimization (including energy and carbon emissions optimization) of the home microgrid under federated learning. The upper circle represents the privacy protection part, which simulates gradient inversion

attacks against federated learning, inferring possible factors likely to lead to privacy leakage.

1.3.1 Smart home part

In this part, a household device action recommendation system is proposed, aiming at building future appliance automation. The user data, including the electricity usage data, appliance location information, appliance usage sequence data, general knowledge of appliance utilization, etc., are recorded, processed, and organized into a knowledge graph. It contains a wealth of information on everyday household appliance usage. Subsequently, a framework called 'DARK' is proposed, encompassing recommendations, interpretations ability of the recommendations, and a demand response algorithm. In this framework, the Knowledge graph attention network (KGAT) algorithm is customized and improved regarding sampling and aggregation compared to the traditional KGAT. To start with, the user's next possible action is recommended with the use of the knowledge graph. Comparison simulations have been conducted with traditional KGAT, Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). The proposed algorithm performance surpasses them. Secondly, an embedding-based interpretation method was designed, this is employed in the analysis of the reasons for each recommendation decision. Lastly, demand response optimization is carried out based on the anticipated recommended actions, considering both power consumption and user expected satisfaction, resulting in the optimized power level settings for each appliance. Energy efficiency is thereby enhanced by balancing the energy consumption with the user expected satisfaction. This part will be introduced in Chapter 3.

1.3.2 Home microgrid part

In this part, a home microgrid scheduling algorithm is proposed. A federated learning system to predict photovoltaic power and load is developed to provide a future possible scheduling environment to support the scheduling algorithm. Based on this

federated prediction system, a Deep Q-Network (DQN) -based scheduling algorithm is developed, using DQN to capture the optimal electricity purchasing timing and to make real-time decisions to reduce the client's cumulative electricity cost and total carbon emissions. Subsequently, to protect a user's privacy, federated learning is used to link the deep learning part of DQN, therefore protecting privacy while fully utilizing the data from each DQN client. The results are compared with those of time-based scheduling, standalone DQN and multi-agent DQN regarding optimization effects, demonstrating the advantages of the proposed scheduling algorithm. Finally, the adjustability of the proposed algorithm is explored, in which the DQN agents in the Pareto front are stored. This allows each client to choose the corresponding agent based on their preferred objectives, that is, selecting between the preference for reducing cumulative electricity prices or lowering carbon emissions. This part will be introduced in Chapter 4.

1.3.3 Privacy protection part

This part proposes a privacy leakage measurement framework for federated home applications. The test environment is established based on: 1) The recommendation system in the Smart home part, and 2) Load forecasting in the Home microgrid part. Afterwards, a framework, called 'SAFE-Home', is proposed to measure the possibility of potential privacy leakage within federated learning. The GS attack is chosen to be simulated in the federated learning environment. The data from the simulated attacks is collected and preprocessed. By leveraging the data collected and integrating explainable machine learning, a new algorithm in the 'SAFE-Home' framework is proposed to explain the key factors for privacy leakage under the gradient inversion algorithm. The 'SAFE-Home' algorithm has two parts: 1. A Long-Short Term Memory (LSTM) part to predict the extent of any privacy leakage. 2. An improved explainable machine learning algorithm to identify the key factors leading to privacy leakage under GS attacks. Finally, strategic advice for privacy protection regarding adjusting the batch size, training iterations, architecture size, activation functions used, and input dimensions is derived from extensive simulations and theoretical analysis. This part will be introduced in Chapter 5.

1.4 Thesis structure

The organization of the thesis is as follows: Chapter 2 introduces the background and literature review, divided into three aspects: Firstly, smart communities encompassing smart homes and home microgrids are introduced. The smart home section mainly introduces device action recommendation systems, while the home microgrid section focuses on scheduling and optimization algorithms. Secondly, privacy protection is focused on, including an introduction to federated learning and its potential privacy risks. Finally, AI is introduced, including deep learning algorithms, recommendation algorithms, deep reinforcement learning algorithms, and the explainability or interpretability of machine learning.

Chapter 3 introduces DARK, a knowledge graph-based recommendation framework for the household device action recommendation system. Initially, the collected device on-off electrical power data is customized and transformed into a knowledge graph. Secondly, an improved KGAT algorithm is proposed for knowledge graph embedding aggregations. Subsequently, an interpretable graph analysis is conducted based on graph embeddings. Lastly, based on the results of the recommendations, demand response is performed to optimize user expected satisfaction and energy consumption, forming a Pareto front.

Chapter 4 introduces the scheduling algorithm in a home microgrid, integrating four homes and renewable energy sources. It utilizes the DQN algorithm to decide on the best opportunity for electricity purchases, whilst considering the current state of the home microgrid. Subsequently, the DQN is combined with federated learning, leveraging the data from each household while protecting their privacy. In addition, the proposed bi-objective optimization approach of DQN stores the model parameters of DQN agents on the Pareto front, allowing clients to choose the appropriate DQN agent based on their optimization preferences.

Building on the foundations of Chapters 3 and 4, Chapter 5 explores the privacy leakage and protection mechanisms of federated learning. It simulates the presence of a malicious attacker on the server during the federated learning process. The attacker utilizes the Gradient Similarity algorithm to recover the original training data, using the differences in the model parameters of federated learning. The chap-

ter proposes a bi-layer structure based on long short-term memory and explainable machine-learning algorithms. Privacy leaks can be predicted, and any vulnerabilities of the model's inputs that may lead to privacy leaks under the Gradient Similarity attack can be identified.

Chapter 6 is the conclusion, including a summary of contributions, innovations and future potential research directions.

CHAPTER 2

Literature review

The background consists of three parts: 1. Smart communities, including an introduction to smart homes and home microgrids. 2. Privacy protection, primarily covering the potential risks in federated learning. 3. Artificial intelligence and machine learning, focusing on AI algorithms.

2.1 Smart communities

Smart communities aim to enhance the residents' quality of life, increasing energy efficiency, as well as promoting environmental sustainability. In this thesis, smart communities can be seen as a combination of small-scale smart grids (smart homes) and microgrids (home microgrids). Smart homes focus on optimization within the household, whereas home microgrids lean towards community power optimization. Combined, these form an efficient, flexible, and sustainable energy ecosystem for community residents.

To simplify and introduce the smart community concept, Fig. 2.1 shows the basic units of a smart community, consisting of a schematic diagram describing a smart home and a home microgrid. The upper half of Fig. 2.1 shows a typical

smart home architecture, where sensors are used for real-time monitoring of the power usage and operational status of the household appliances. This monitoring data is transmitted to a server and an energy management system, where algorithms optimize the smart homes' operation and energy usage, with feedback for decision-making. The key features of the smart home include:

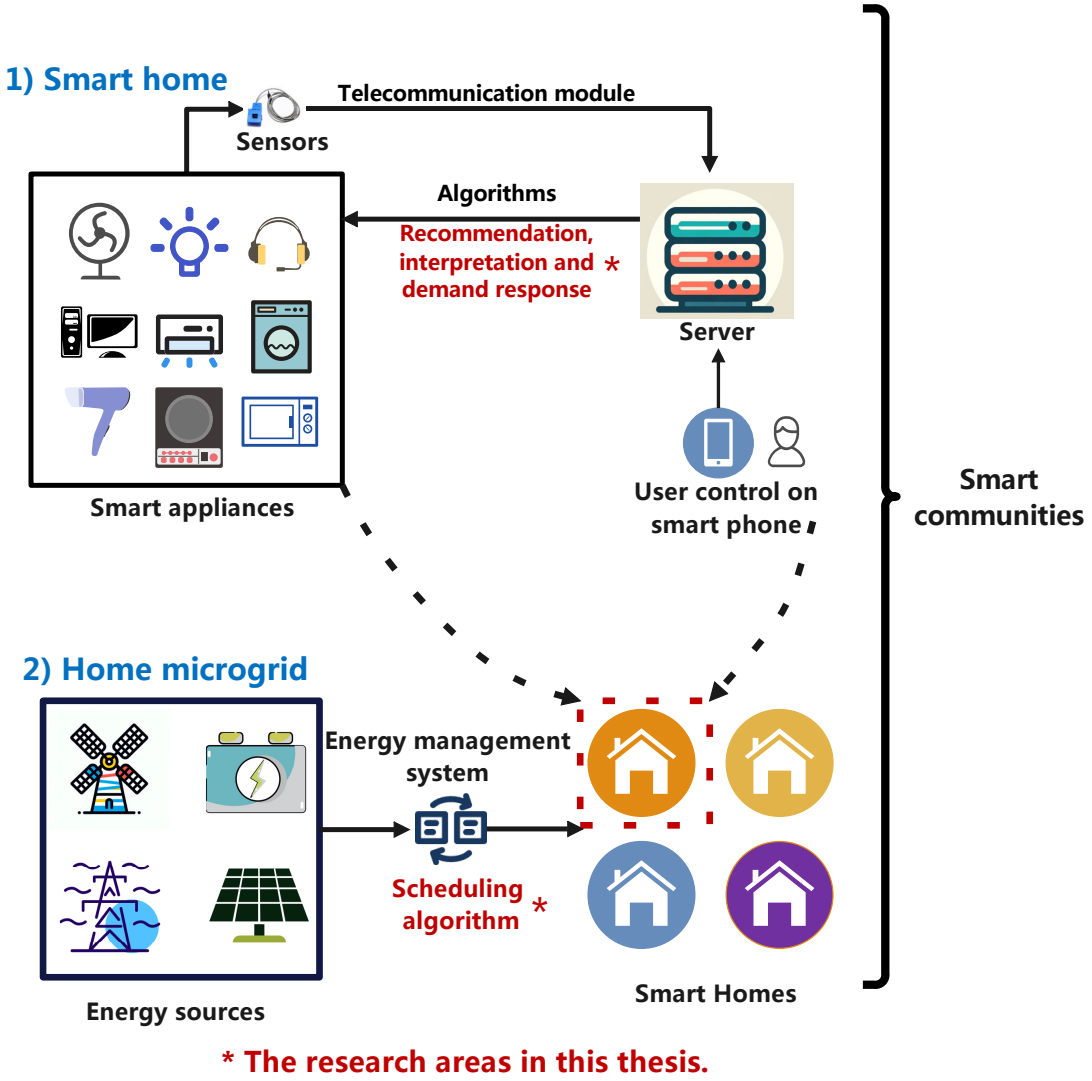

Figure 2.1: Schematic diagram of a smart community, including 1) A smart home, mainly processing household appliances data information. and 2) A home microgrid, mainly processing the small-scale power information of several households.

1. Automated control: Including security monitoring, household action recommendation systems, and home heating system control.

2. Energy management system: Including developing algorithms for hardware

devices, power and demand response algorithms, and improving energy efficiency.

3. Telecommunication and remote control: Including the development of home IoT architecture and algorithms for improving remote control.

4. Environmental comfort: Including the improvement of the level of comfort.

Smart home development and research are crucial for better living conditions, energy savings, enhanced security, household convenience as well as long term sustainability [11] [12].

Home microgrids primarily focus on energy optimization within a community of a few households. A typical home microgrid architecture is shown in the lower half of Fig. 2.1, including renewable energy resources, an energy storage system, the main power grid, etc. An energy management system is used to control the overall energy operations. The key features of the home microgrid include [13]:

1. Distributed energy resources: Including increasing the penetration of renewable energy and developing prediction algorithms.

2. Energy storage systems: Developing energy efficient storage algorithms and scheduling algorithms.

3. Energy sharing and trading: Developing smart trading algorithms.

4. Enhanced resilience: Enhancing the reliability of home microgrids.

Similar to microgrids, home microgrids aim to increase the penetration of renewable energy, to protect power trading among users, to enhance energy efficiency, while reducing carbon emissions [14].

In microgrids, the distance between homes affects both the energy transmission and communication efficiency. For example, compared to houses, flats often have higher centralized control efficiency. In this thesis, it is assumed that all energy transmission and communication are ideal, and related issues will not be considered in depth. The main focus will be on the algorithms.

This thesis proposes a smart home recommendation algorithm in Chapter 3 and develops a home microgrid optimization algorithm in Chapter 4, the related literature review will be introduced in sub-section 2.1.1 and 2.1.2.

2.1.1 Smart home household recommendation systems

With the advent of smart home appliances, there has been a growing interest in replacing conventional automation with intelligent action recommendation systems [5], [15]. Accurate action recommendations can enhance the convenience of daily life, such as by automatically turning on the dining area lights during meals, closing curtains at night, or turning on and off the TV/music based on residential habits. For the elderly or people with mobility difficulties, automatically controlling appliances can also improve the quality of their lives [16].

This sub-section focuses on reviewing household device action recommendation systems. Research in this area is limited and is still in the preliminary stages of development. All the related papers introduced in this section are summarized in table 2.1 and 2.2. The detail of recommendation algorithms will be introduced in sub-section 2.3.2.

In 2014, Rasch, Katharina *et al.* proposed a smart home recommendation system, which continuously read the user's current situation and recommended services that aligned with the user's habits, aiming to address the issues of complicated user interfaces and difficult operations in smart home applications [17]. An unsupervised machine learning algorithm was used for recommendations. However, due to the limitations of the algorithms, its accuracy was only about 60%.

In 2016, Chen, Hao *et al.* proposed a weighted hybrid recommendation system based on the Kalman Filter model to predict the next possible actions of users [18]. The method that was proposed combined contextual filtering, collaborative filtering and content-based recommendations. Similar to the previous algorithm, this algorithm could also not achieve a high recommendation accuracy, with the best recall value of 0.82 only achievable when the number of recommendations were 10. When the number of recommendations were 2, the recall rate could only reach an accuracy of 0.57. Please note that 'the number of recommendations' describes the comparison

of the top-selected recommendation data with the actual label. For example, if the number of recommendations is 10, it means taking the top 10 recommended items and comparing them with the selected label. If the label is among these values, then the recommendation is accurate. Therefore, increasing the number of recommendations will improve the accuracy because there are more possible results to be matched with each label.

In 2016, Belghini, Naouar *et al.* proposed a smart home recommendation system, which offered personalized services using contextual information and physical sensor data [19]. It activated the most appropriate services based on the user's current activity, time, location, temperature, and special events. The major drawback of the algorithm was that it primarily focused on the theoretical framework and methodology without providing detailed descriptions of the experiments or results.

In 2021, Reyes-Campos, Josimar *et al.* proposed a method for discovering resident behaviour patterns using machine learning techniques and the Internet of Things technologies [20]. A smart home control platform was proposed, utilizing the C4.5 algorithm to generate decision trees and capture the daily activity patterns of residents. However, this work didn't consider energy optimization as well as the interpretability of the algorithm.

In 2022, Jeon, Hyunsik *et al.* proposed an action recommendation method for smart homes [21]; each device action was summarized through a self-attention mechanism, and the user's patterns sequence was extracted using a query-attention mechanism. The recommendation algorithm was implemented on a customized dataset. Nevertheless, the dataset used by the algorithm was specially customized and non-continuous, limiting its applicability in other datasets. Moreover, it failed to consider the use of an energy optimization method based on the recommended results.

In 2022, Osman, Krešimir *et al.* presented a PID-based building system. The control algorithm was developed to automatically set the control parameters of the building, aiming to reduce thermal energy demand [22]. Different operating modes were set for the building system based on usage time.

In 2022, Varlamis, Iraklis *et al.* proposed a recommendation system that integrated sensor data, user habits, and user feedback to provide personalized energy-

Table 2.1: Literature review on household device action recommendation systems, table 1.

Work (brief description)	Based on	Energy optimization	Personalization	Contextual awareness	User satisfaction model	Location	Interpretability
Wenzhi Chen *et al.* (2024) proposed a knowledge graph-based algorithm to make device action recommendations with interpretability, while considering the demand response. (It will be shown in Chapter 3)	Knowledge graph attention network	Yes	Yes	Yes	Yes	Yes	Yes
Rasch, Katharina *et al.* (2014) proposed an unsupervised recommendation algorithm based on the user's habits. [17]	A type of unsupervised learning	No	Yes	Yes	No	Yes	No
Chen, Hao *et al.* (2016) proposed a weighted hybridization Kalman filter model to predict the user's next action in the smart home. [18]	Kalman Filter	No	Yes	Yes	No	No	No
Belghini, Naouar *et al.* (2016), proposed a recommendation algorithm using contextual information and physical sensor data. [19]	Neural network	No	Yes	Yes	Yes	Yes	No
Reyes-Campos, Josimar *et al.* (2021) proposed a method for discovering resident behaviour patterns using machine learning and the Internet of Things technology. [20]	C4.5	No	Yes	Yes	Yes	Yes	No
Jeon, Hyunsik *et al.* (2022) proposed an action recommendation method for smart homes, summarizing the device control through a self-attention mechanism. [21].	Transformer	No	Yes	Yes	No	Yes	No

Table 2.2: Literature review on household device action recommendation systems, table 2.

Work (brief description)	Based on	Energy optimization	Personalization	Contextual awareness	User satisfaction model	Location	Interpretability
Osman, Krešimir *et al.* (2022) presented a method for applying a control system in a building, including theoretical models and PID control simulations, and a Siemens Portal-based remote system was used to reduce thermal energy needs. [22]	Siemens Portal	Yes	No	No	Yes	Yes	No
Varlamis, Iraklis *et al.* (2022) integrated sensor data, user habits, and feedback with the EM3 platform to provide personalized energy recommendations. [23]	EM3 platform and long-short term memory	Yes	Yes	Yes	No	Yes	No
Yuhang Yao *et al.* (2023) created a unique graph for each user and made recommendations. He used federated learning to protect privacy. [24]	GrphSAGE	Yes	Yes	Yes	No	No	No
Ali, SM Murad *et al.* (2023) developed personalized automation systems with transfer learning that provided smart service components to users and optimized algorithms with feedback. [25]	Transfer learning	No	Yes	Yes	No	Yes	No
Dilekh, Tahar *et al.* (2024) proposed a dynamic, context-aware recommendation system for smart homes with a combination of supervised and unsupervised learning. [26]	FP-growth algorithm and generalized linear model	Yes	Yes	Yes	Yes	Yes	No

saving suggestions [23]. With the C4.5 algorithm and Weka API, the system acquired the best time to make energy-saving recommendations, such as by adjusting thermostat settings or by turning off any unnecessary lights. However, this work only applied to specific scenarios, such as predicting unnecessary lighting or air conditioning use, which limited its applicability in practical settings.

In 2023, Yuhang Yao *et al.* developed a recommendation system using GraphSAGE [24]. This system made recommendations by creating a unique graph for each user. Both the proposed approach in Chapter 3 of this thesis and [24] utilize graph-based recommendation algorithms. The difference is that in the proposed recommendation system in Chapter 3, the actions are triggered by electricity usage, leading to demand response-driven optimization. Additionally, the algorithm proposed in Chapter 3 is based on a knowledge graph which can incorporate information from the edges of the graph, yet this was not considered in [24].

In 2023, Ali, SM Murad *et al.* developed personalized automation systems for smart homes [25]. Transfer learning was used for the training and making recommendations. Similarly, the work didn't consider the interpretability of the algorithm and further energy optimization.

In 2024, Dilekh, Tahar *et al.* proposed a dynamic, context-aware recommendation system for smart homes [26]. The unsupervised FP-growth algorithm and the supervised generalized linear model, were combined in this system to enhance home automation. The work didn't consider the interpretability of the system's decisions.

According to the comparison in table 2.1 and 2.2, only the proposed algorithm in Chapter 3 (the first item in table 2.1) considered all the factors, including energy optimization, personalized recommendations, user satisfaction model, interpretability, as well as incorporating contextual awareness.

2.1.2 Scheduling algorithm in microgrids with DRL

Deep reinforcement learning has been increasingly utilized in microgrid scheduling. This involved optimizing the grid operation, integrating distributed energy resources, and balancing supply and demand in real-time to enhance the efficiency and sustainability of microgrids.

This sub-section focuses on reviewing home microgrid scheduling algorithm with DRL algorithms. All the related papers introduced in this section are summarized in table 2.3, 2.4 and 2.5. The detail of deep reinforcement learning algorithms will be introduced in sub-section 2.3.3.

In 2018, Mocanu, Elena *et al.* proposed the first paper that used deep reinforcement learning in smart grids for online optimization in building energy management systems [28]. Deep Q-learning and deep policy gradient methods were utilized, and both algorithms were expanded to execute multiple actions simultaneously, optimizing energy costs on the demand side.

In 2020, Wang, Biao *et al.* proposed a deep reinforcement learning method for optimizing the interruptible load on the demand side [29]. The demand response architecture was constructed and expressed as a Markov decision process, and the Dueling deep Q Network was used to reduce the power grid's peak load demand and operational costs.

In 2020, Chung, Hwei-Ming *et al.* proposed a model-free approach for managing household power consumption, both saving on electricity bills and reducing the load during peak times [30]. The interaction between the households and the power grid was modelled as a non-cooperative stochastic game, and the distributed deep reinforcement learning algorithm was used to find the game's Nash equilibrium.

In 2020, Liu, Yuankun *et al.* proposed a home energy management optimization algorithm based on Deep Q-learning and Double deep Q-learning [31]. These algorithms helped users reduce electricity consumption by responding to a dynamic environment. Tests showed that Double deep Q-learning was more effective than DQN in reducing Home Energy Management System costs.

In 2020, Ye, Yujian *et al.* proposed an energy management strategy for residential multi-energy systems based on the DDPG [32]. Compared to the traditional stochastic programming method, this strategy reduced energy costs further.

In 2021, Nakabi, Taha Abdelhalim *et al.* proposed an energy management system [35]. A microgrid model was developed, including wind turbines, an energy storage system, adjustable loads, price-responsive loads, and a connection to the main grid. This work implemented and compared seven deep reinforcement learning algorithms

Table 2.3: Literature review on microgrid scheduling algorithms with deep reinforcement learning, table 1.

Work (brief description)	Based on	Energy optimization	Carbon optimization	Model adjustable	Privacy protection
Wenzhi Chen *et al.* (2022) proposed a energy and carbon optimization system for home microgrids as well as considering privacy protection with an improved deep reinforcement learning algorithm.	Deep Q-network	Yes	Yes	Yes	Yes
Atallah, Ribal F *et al.* (2018) proposed a model based on deep Q-networks to ensure the service and safety of vehicles. [27]	Deep Q-network	Yes	No	No	No
Mocanu, Elena *et al.* (2018) achieved online optimization for building energy management systems in building system, electricity vehicles and PV panels. [28]	Deep-Q network	Yes	No	No	No
Wang, Biao *et al.* (2020) optimized the interruptible load in demand response, regulating to ensure the voltage safety and reducing the load demand and costs. [29]	Dueling deep-Q network	Yes	No	No	No
Chung, Hwei-Ming *et al.* (2020) proposed an algorithm to solve the stochastic game between the household and the power grid. [30]	Actor-Critic	Yes	No	No	Yes
Liu, Yuankun *et al.* (2020) proposed a home energy management optimization strategy to reduce electricity consumption in a dynamic environment. [31]	Double deep-Q network	Yes	No	No	No

Table 2.4: literature review on micrigrid scheduling algorithms with deep reinforcement learning, table 2.

Work (brief description)	Based on	Energy optimization	Carbon optimization	Model adjustable	Privacy protection
Ye, Yujian *et al.* (2020) proposed a real-time autonomous energy management strategy for residential multi-energy systems. [32]	DDPG	Yes	No	No	No
Bahrami, Shahab *et al.* (2020) proposed a demand response algorithm, considering the uncertainties, privacy, and power constraints. [33]	Actor-Critic	Yes	No	No	Yes
Yang, Ting *et al.* (2021) proposed a model-free energy dispatch strategy to optimize the operation of integrated energy systems. [34]	DDPG	Yes	No	No	No
Nakabi, Taha Abdelhalim *et al.* (2021) used seven deep reinforcement learning algorithms in microgrid energy management. [35]	Seven reinforcement learning algorithms	Yes	No	Yes	No
Zhao, Liyuan *et al.* (2022) proposed a scheduling strategy for multi-energy systems to minimize residents' energy costs while considering power uncertainties. [36]	PPO	Yes	Yes	No	No
Wang, Jianing *et al.* (2022) proposed a game model for a virtual power plant integrated with electric vehicle charging stations, making them stable and efficient. [37]	Delay DDPG	Yes	No	No	No
Ren, Mifeng *et al.* (2022) proposed a prediction-based optimization method for the home energy management system. [38]	Duling deep-Q network	Yes	No	No	No

Table 2.5: Literature review on micrigrid scheduling algorithms with deep reinforcement learning, table 3.

Work (brief description)	Based on	Energy optimization	Carbon optimization	Model adjustable	Privacy protection
Wang, Kang *et al.* (2022) proposed a model to control the charging behaviour of vehicles, optimizing electricity pricing. [39]	DDPG	Yes	No	No	No
Alqahtani, Mohammed *et al.* (2022) proposed a reinforcement learning model for electricity vehicles to address the uncertainties in demand response and to reduce energy costs. [40]	Deep Q-networks	Yes	No	No	No
Zhang, Yang *et al.* (2022) proposed a multi-agent method for optimizing energy purchase strategies for electric vehicles, charging stations and power grids. [41]	Multi-agent DDPG	Yes	No	No	No
Lu, Yu *et al.* (2023) proposed a multi-agent scheduling algorithm with designed reward function for active distribution systems, considering the uncertainties. [42]	Multi-agent DDPG	Yes	No	No	No
Liu, Ding *et al.* (2023) proposed a model for electric vehicle scheduling and voltage control algorithms, aiming to reduce costs and maintain voltage stability. [43]	DDPG	Yes	No	No	No
Yang, Yanhong *et al.* (2023) proposed an energy strategy for microgrids focused on cost efficiency, emissions reduction, and privacy protection. [44]	Improved double deep-Q network	Yes	Yes	No	Yes

to minimize electricity costs, while making the models adjustable.

In 2021, Yang, Ting *et al.* proposed a model-free dynamic energy dispatch strategy based on an improved DDPG algorithm, optimizing the operation of Integrated Energy Systems [34]. Simulation results demonstrated that this strategy had faster convergence and lower operating costs when compared to the original DDPG strategy.

In 2022, Zhao, Liyuan *et al.* proposed a joint load scheduling strategy for household multi-energy systems to minimize residents' energy costs while maintaining thermal comfort [36]. By utilizing deep reinforcement learning, the system was able to simultaneously control both the continuous actions of the power-shiftable devices and the discrete actions of the time-shiftable devices.

In 2022, Wang, Jianing *et al.* proposed a bi-layer scheduling method for virtual power planet based on deep reinforcement learning [37]. It considered the uncertainty of renewable energy and established a scheduling framework for day-ahead and intra-day operations. The implementation scheme for both price-based and incentive-based demand response for flexible loads was determined.

In 2022, Ren, Mifeng *et al.* proposed a prediction-based optimization method for real-time scheduling of the home energy management system, utilizing a deep reinforcement learning approach for optimal dispatch of the power flow [38]. The simulation demonstrated that this method reduced user costs while maintaining user satisfaction.

In 2023, Lu, Yu *et al.* proposed a multi-agent deep reinforcement learning-based algorithm for the real-time optimal scheduling in distribution systems, while at the same time considering the uncertainties in renewable generation, loads, and electricity prices [42]. The electricity costs were optimized, and the algorithm was adaptable to uncertainties.

Many other studies have explored the use of advanced Deep Reinforcement Learning algorithms to optimize energy management systems [27] [33] [40] [39] [39] [37] [41] [43] [44]. According to the comparison in table 2.3, 2.4 and 2.5, only the proposed algorithm in Chapter 4 (the first item in table 2.3) considers all the factors, including energy and carbon optimization, an adjustable model for different

objectives and privacy protection.

2.1.3 Demand response

Demand response (DR) is a strategy aimed at encouraging consumers to adjust or reduce their electricity usage during specific periods (such as peak hours) in response to changes in power generation. The main purposes and features of demand response include [45] [46]:

1. Balancing power supply and demand: Demand response helps maintain a demand-supply balance by reducing or delaying some demand.

2. Reducing electricity costs: Demand response decreases reliance on expensive emergency power sources and lowers overall electricity costs.

3. Increasing grid reliability: By managing peak demand, demand response helps reduce the risk of grid overload, thus enhancing the power grid's reliability.

4. Supporting the integration of renewable energy: Demand response allows flexible adaptation to renewable energy sources like solar and wind, promoting their penetration.

Demand response has been the focus of many research papers. In 2019, Pallonetto, Fabiano *et al.* implemented demand response control algorithms in the residential sector using predictive algorithms. Comprehensive instrumentation tests were conducted in a typical house that represented Ireland's most common building type. A calibrated building simulation model was developed to assess the effectiveness of demand response strategies under various time-of-use electricity tariffs and zone thermal controls [47]. In 2020, Li, Hepeng *et al.* proposed a real-time demand response strategy for the optimal scheduling of home appliances based on deep reinforcement learning. This strategy considered the uncertainties in resident behavior, real-time electricity prices, and outdoor temperatures. It can handle both discrete and continuous actions to optimize the scheduling of various types of appliances. Its effectiveness has been validated through simulation [48]. In 2020, Alfaverh, Fayiz *et al.* proposed an effective home energy management system for

demand response using reinforcement learning and fuzzy reasoning. This system optimized the scheduling of smart home appliances through the Q-learning algorithm, shifting their operations from peak to off-peak periods. Simulation results showed that this method can smooth both the power consumption curve and minimize electricity costs while considering the user's preferences [49]. In 2021, Duman, A Can *et al.* proposed a mixed-integer linear programming-based home energy management system for scheduling the load a day ahead to reduce costs, achieve optimal demand response and self-consumption of photovoltaic energy, while efficiently managing air conditioning demand response and maintaining thermal comfort through a fuzzy logic-based thermostat. Simulations demonstrated a reduction in both daily and air conditioning costs [50]. In 2022, Chen, Zhe *et al.* proposed a demand response scheduling method for residential buildings, employing the Nondominated Sorting Genetic Algorithm II as a multi-objective optimization algorithm to minimize the electricity costs and inconvenience index [51]. The proposed scheduling method was evaluated on working and non-working days. It effectively shifted the peak load to off-peak periods, reducing electricity bills, and meeting residents' comfort needs. In 2024, Wen, Lulu *et al.* proposed a dynamic price-based demand response model for demand side management in smart grids [52]. This model can shift peak electricity demand, thereby enhancing the stability and reliability of the power system. Through the proposed demand response model, the peak electricity demands of commercial and residential electricity consumers decreased by 4.99% and 9.99%, respectively, while maintaining the interests of the electricity consumers.

In this thesis, demand response is focused on balancing a resident's satisfaction with power consumption by setting the appropriate power level for each appliance, thus improving overall energy efficiency. Some recent research in demand response algorithms are summarized in the table 2.6.

2.2 Privacy protection

Nowadays, data is no longer confined to the purpose of training but exists more as a form and essence of an asset. Data is crucial for governments, businesses, and

Table 2.6: Literature review on demand response applications in recent years.

Year	Reference	Based on	Contribution
2019	Pallonetto, Fabiano *et al.* [47]	Linear regression	Compared the performance of rule and predictive-based demand response algorithms in a typical Irish residential dataset, achieving cost and carbon emissions reductions.
2019	Houman*et al.* [53]	Mixed integer nonlinear programming	A hybrid price-based demand response strategy was proposed, considering the uncertainty of the decision variables and parameters in the microgrid, implementing day-ahead scheduling.
2020	Li, Hepeng *et al.* [48]	Trust region policy optimization	Proposed a demand response strategy based on the home appliances' scheduling, considering the uncertainties of residents behaviour, electricity prices, and temperature.
2020	Alfaverh, Fayiz *et al.* [49]	Fuzzy logic and Q-learning	An residential demand response algorithm was proposed, aiming to reduce electricity costs and smooth the power curve while considering user preferences.
2021	Duman, A Can *et al.* [50]	Mixed-integer linear programming	Proposed a system, combining smart thermostats with home energy management systems, effectively reducing daily costs through linear programming and fuzzy logic.
2022	Chen, Zhe *et al.* [51]	Nondominated sorting genetic algorithm II	A demand response approach for a building was proposed on four kinds of loads, effectively shifting peak loads to off-peak periods, reducing electricity costs, and ensuring comfort.
2022	Tostado-Véliz, Marcos *et al.* [54]	Four Strategies	A home energy management system with three demand response strategies were developed, improving key indicators by up to 70% with minimal impact on electricity costs.
2022	Stanelyte, Daiva *et al.* [55]	Overview	Discussed how the electricity industry can enhance transmission system efficiency through demand response, with the emphasis of IoT and blockchain technologies.
2022	Sharda, Swati *et al.* [56]	Dynamic itemset counting	A consumer behaviour model was developed for demand response, combining power forecasting by CNN-LSTM and appliance association mining to improve response effectiveness.
2022	Shreenidhi, HS *et al.* [57]	Modified elephant herd optimization algorithm	A load scheduling optimization algorithm for home energy management systems was proposed to reduce electricity costs, minimize peak-average ratio, and enhance user comfort.
2022	Yu, Biying *et al.* [58]	Non-dominated sorting genetic algorithm	Explored the potential changes in peak-to-valley electricity consumption and the costs made by participating in power demand response.
2022	Blaschke, Maximilian J *et al.* [59]	Economic simulation	Proposed a algorithm that reforms electricity taxation dynamically according to trading prices, and that makes residential demand-side management profitable.
2023	Huang, Jueru *et al.* [60]	Multi-agent reinforcement learning	An hour-ahead demand response algorithm for home energy management was proposed, utilizing neural networks and multi-agent reinforcement learning.
2024	Wen, Lulu *et al.* [52]	Game theory model	A dynamic price-based demand response game theory model for smart grids was proposed, aiming to enhance power system stability by adjusting peak electricity demand.

individuals. In Europe, two privacy protection regulations affect the UK and EU members: the General Data Protection Regulation and the UK's Data Protection Act [6] [7]. General Data Protection Regulation, effective May 25, 2018, applies to all member states within the European Economic Area. It mandates that businesses obtain explicit user consent for private data and provide transparent privacy policies. Violating regulation provisions can lead to hefty fines, up to 4% of the company's global annual revenue or 20 million Euros, whichever is higher. Post-Brexit, the UK enacted the Data Protection Act, which was harmonized with the General Data Protection Regulation and applies to the UK. It also protects users' data and regulates data controllers and processors.

2.2.1 Privacy protection in energy systems

In energy systems, there is much research on privacy protection technologies. These common technologies include battery-based load hiding, differential privacy, encryption and anonymization, and federated learning [61].

Battery-based load hiding

Battery-based load hiding is a technology that uses rechargeable batteries to obscure the users' electricity usage patterns, preventing privacy leaks to power companies and third-party data analysis organizations. In 2019, Hossain, Mohammad Belayet *et al.* proposed a method that used rechargeable batteries to enhance the privacy protection of smart meters, while optimizing the target output load over time through the use of the artificial fish swarm optimization algorithm. Both offline and online privacy protection mechanisms were considered, effectively reducing the energy consumption costs as well as protecting privacy [62]. In 2019, Jiajia, Xu *et al.* proposed a method that combined solar power generation systems with rechargeable batteries to protect the privacy of smart meters. A 'privacy-safe and cost-friendly optimization model' was developed, aiming to minimize the weighted sum of any privacy leakage and financial cost [63]. In 2020, Bovornkeeratiroj, Phuthipong *et al.* proposed RepEL, a rule-based privacy protection algorithm, to determine 'replay records' based on the current charging status of the battery, thereby obscuring the

user's behavior information and protecting the privacy of the smart meter's data. Simulations showed that this protection could limit the privacy leakage rate to below 10% [64]. Other related algorithms can be found in [65] (written by Desai, Sanket *et al.* in 2019), which is a survey of battery-based load hiding.

Differential privacy

In 2006, Dwork, Cynthia *et al.* proposed the conception of differential privacy [66], a technique for protecting individual privacy during data publication, which involved adding a certain amount of noise to the data without significantly altering the output results. This method was widely used in the privacy protection of machine learning, ensuring that the results of the data analysis retain the statistical properties while not revealing specific personal information. In 2019, Fioretto, Ferdinando *et al.* proposed a differential privacy-based method for publishing power grid data, aiming to hide the parameters of the transmission lines and transformers. Simulations demonstrated that the proposed algorithm maintained the feasibility of the power grid in the AC power flow and significantly reduced the damage from potential attacks [67]. In 2019, Hassan, Muneeb Ul *et al.* proposed a differential privacy-based real-time load monitoring algorithm that protected the smart meter users' privacy by adding noise and limiting the peak values, while also integrating renewable energy resources [68]. In 2021, Liu, Xing *et al.* proposed a battery-based intermittent differential privacy scheme that generated noise to protect the smart meter's data privacy through charging and discharging of the battery. Additionally, it utilized a reinforcement learning algorithm to optimize the battery control strategies, achieving the dual goals of privacy protection and cost savings. Other related algorithms can be found in [68] (written by Hassan, Muneeb Ul *et al.* in 2019), which is a survey of differential privacy for cyber physical systems.

Encryption and anonymisation

Encryption and anonymization are commonly used in various research areas. In 1996, Canetti, Ran *et al.* proposed the secure multi-party computation [69], a cryptographic technology that allowed multiple parties to jointly compute a function

without revealing their private data to each other. This technique enabled the sharing of computational results while keeping the input data of each participant confidential, thus protecting data privacy. In 2002, Sweeney, Latanya proposed a privacy protection model called 'k-anonymity' [70]. A set of deployment policies were introduced to release data while ensuring scientific validity. K-anonymity ensured that in the released data, the information of each individual could not be distinguished from that of at least k-1 (k minus 1) other individuals. In 2009, Gentry, Craig proposed Homomorphic encryption, a cryptographic technique that allowed for computations to be performed directly on encrypted data without needing to decrypt it. This means complex computational operations can be executed while maintaining data privacy, which is particularly important for cloud computing and secure data analysis. Homomorphic encryption enabled data owners to share encrypted data, allowing third parties to process and analyze it without access to the original, unencrypted data [71].

There are also many applications in power systems. In 2020, Kong, Wei *et al.* proposed a practical group blind signature scheme to achieve privacy protection in smart grids. By using homomorphic encryption and group blind signatures, the scheme not only provided anonymous authentication and data integrity verification but also allowed tracking of malicious user identities when necessary [72]. In 2021, Baza, Mohamed *et al.* proposed privacy-preserving and collusion-resistant charging coordination schemes. Based on anonymous tokens, these schemes effectively protected the privacy of the energy storage unit owners and prevented link-ability attacks through centralized and decentralized methods [73]. In 2022, Tran, Hong-Yen *et al.* proposed a privacy-preserving scheme without the need for a trusted third party. By combining perturbation and encryption techniques, it protected the privacy of high-frequency consumption data from smart meters while maintaining the accuracy and efficiency of the energy services. Other related algorithms can be found in [74] (written by Abreu, Zita *et al.* in 2022) and [61] (written by Mirzaee, Parya Haji *et al.* in 2022), which are surveys of privacy protection for power systems.

Since federated learning is the focus of this thesis, its related review is summarized in the next section, Section 2.2.2 and 2.2.3.

2.2.2 Federated learning

Due to strict privacy laws and regulations, different data management and storage standards, and user privacy concerns, there is a 'data island' issue, where deep learning models cannot fully utilize data from various sources, hindering improving the effectiveness of algorithms. A distributed machine learning approach is needed to make each data-holding institution train models independently using their data and then interact with other models. This is the motivation behind the proposal of federated learning.

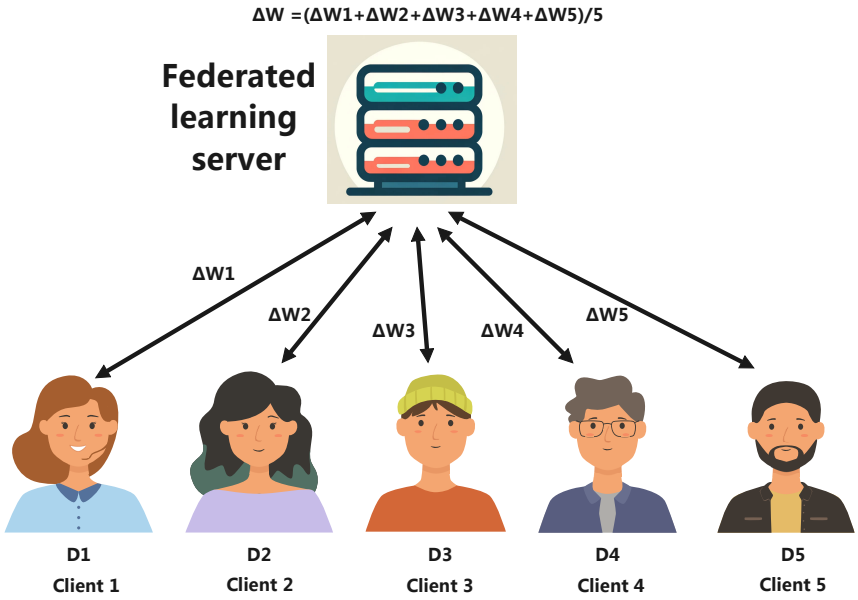

Figure 2.2: A diagram of federated learning. Instead of uploading the private data, the model parameters are uploaded to the server.

Federated learning is a distributed machine learning method that allows for collaborative training and data usage while ensuring that data remains local [75]. In 2015, H. Brendan McMahan *et al.* proposed FedAvg, which was the first paper of federated learning [75]. In FedAvg, instead of uploading the original data, the neural network weights were uploaded from clients to the server after local training (the detail of deep learning will be introduced in sub-section 2.3.1). The data of each client were used by the server on the premise of protecting privacy. This led to three advantages. Firstly, storing large amounts of data on the server was no longer necessary. Secondly, the server used each client's data while protecting privacy. Finally, the computing was pushed to the edge, reducing the communication rounds

in the network. Based on FedAvg, many federated learning models were proposed. Some of the well-known algorithms include Federated Learning with Proximal Term (proposed by Tian Li *et al.*, 2020) [76], Federated Learning with Matched Average (proposed by Hongyi Wan *et al.*, 2020) [77], and Q-FedAvg (proposed by Tian Li *et al.*, 2020) [78]. Numerous algorithms were proposed in federated learning, mainly concentrating on improving communication efficiency, handling data heterogeneity, enhancing privacy protection, and increasing model performance as well as generalization capabilities. Assuming data holders are Clients and the data centre is the Server, the general steps of the Federated Averaging (FedAvg) algorithm are as follows:

1. Client model initialization: Each client's model is initialized with a uniform network structure with parameters w_i, i represents the index of the clients, then each clients trained locally using their data,

$$w_i(n) = w_i(n-1) - \eta_{lr} \nabla L(w_i(n)) \ , \tag{2.1}$$

   where w_i is the parameter of the client i, η_{lr} is the learning rate and $\eta_{lr} \nabla L(w_i(n))$ is the batch gradient.

2. Parameter uploading: Clients upload their model parameters $w_i(n)$ to the server.

3. Averaging operation: The server performs an averaging operation on the received parameters.

$$w(n) \leftarrow \frac{1}{I} \sum_{i=1}^{I} w_i(n) \ , \tag{2.2}$$

   where w is the parameter of global model, in total there are I clients.

4. Model update: The server sends the aggregated model back to the clients, and the clients update their local models.

$$w_i(n) \leftarrow w(n). \tag{2.3}$$

Simulations have shown that while protecting privacy, federated learning can closely approximate the ideal model, i.e., the performance of traditional distributed machine learning models with shared data, and usually outperforms models trained solely on local data.

2.2.3 Federated learning in grid applications

Federated learning is widely applied in various real-world scenarios, such as banking, healthcare, etc [79] [80]. This thesis primarily focuses on the application of federated learning in smart communities, especially microgrids. Microgrids contain significant private data, such as load, power transactions, and usage data. Federated learning involves collaborative learning from multiple data sources without directly sharing data, enhancing the power grid's reliability [81]. Various applications of federated learning in microgrids are summarized in table 2.7, 2.8 and 2.9, including federated smart trading, federated predictions, federated scheduling and optimizations as well as federated stability, and security in microgrid.

Federated smart trading

In 2022, Bouachir, Ouns *et al.* proposed Federated Grids [84], an innovative peer-to-peer energy trading and sharing platform that utilized blockchain and federated learning technologies for efficient energy management among microgrids. This platform not only facilitated energy trading but also supported energy sharing, aiming to provide participants with the dual benefits of reduced energy costs and enhanced grid load management. In 2022, Otoum, Safa *et al.* proposed a cooperative and distributed framework based on blockchain and federated learning, aimed at enabling secure energy trading, remote monitoring, and network trustworthiness through the computing, communication, and intelligence capabilities of edge and end devices [85].

Other recent research papers have also focused on federated smart trading [97] [108] [109] [110], where federated learning is commonly used for load and energy consumption prediction. This approach was often combined with blockchain technology, which offered a decentralized and tamper-proof record-keeping system to ensure the security of system transactions.

Table 2.7: Federated learning applications in grids, table 1.

Year	Reference	Field	Based on	Contribution
2020	Zhang, Xiaoning *et al.* [82]	Predictions	Bayesian Long-short term memory	Proposed a federated solar irradiation forecasting scheme based on deep learning, variational Bayesian inference, and federated learning, exhibiting performance advantages.
2022	Gholizadeh, Nastaran *et al.* [83]	Predictions	Long-short term memory	Evaluated the performance of federated learning for short-term forecasting, then further proposed a client clustering method to shorten the convergence training iterations of federated learning.
2022	Bouachir, Ouns *et al.* [84]	Smart trading	Blockchain	Proposed a blockchain and federated learning-based P2P energy trading platform that reduced energy costs for consumers and the load consumption on public utility grids.
2022	Otoum, Safa *et al.* [85]	Smart trading	Blockchain	Proposed a blockchain and federated learning-based framework to enable secure energy trading and remote monitoring.
2022	Wang, Zhenyi *et al.* [86]	Scheduling and optimizations	Transfer learning	Proposed a privacy-preserving framework combining federated learning and transfer learning to evaluate the regulation capacity of HVAC systems in heterogeneous buildings.
2022	Lin, Jun *et al.* [87]	Stability and security	Attention-based convolutional neural network	A federated learning and attention-based convolutional neural network was proposed for diagnosing fault types in power transformers.
2022	Li, Yang *et al.* [88]	Stability and security	Transformer	A federated learning based false data injection attack detection method was proposed, integrating electrical transformers, federated learning, and the Paillier cryptosystem.
2022	Zhang, Yu *et al.* [89]	Scheduling and optimizations	Convolutional neural network	Proposed FedNILM, a practical federated learning paradigm for non-intrusive load monitoring applications.
2022	Lin, Wen-Ting *et al.* [90]	Stability and security	False Data Injection detection	A edge-based federated learning framework was proposed for false data injection attack detection.

Table 2.8: Federated learning applications in grids, table 2.

Year	Reference	Field	Based on	Contribution
2022	Venkataramanan, Venkatesh *et al.* [91]	Prediction	Deep neural network	Proposed a federated learning model for distributed energy resource forecasting using the Internet of Things.
2022	Fekri, Mohammad Navid *et al.* [92]	Prediction	Long short term memory	Used FedAVG and FedSGD for load forecasting, achieving high accuracy and reduced communication rounds.
2022	Moayyed, Hamed *et al.* [93]	Prediction	Convolutional neural network	Proposed a short-term wind power forecasting algorithm with federated learning and convolutional neural networks.
2022	Badr, Mahmoud M *et al.* [94]	Prediction	Long short term memory	Proposed a federated learning based energy forecasting model for metering systems combining an effective data aggregation scheme.
2022	Duan, Zhuoxi *et al.* [95]	Scheduling and optimizations	Double Deep Q Learning	A federated deep reinforcement learning scheduling algorithm was proposed for distributed microgrid, aimed at maximizing the long-term utility of the system.
2022	Du, Yuhan *et al.* [96]	Scheduling and optimizations	Long short term memory	Proposed a federated learning-assisted distributed energy management system in virtual power plants, effectively increasing the performance in power demand forecasting and improving the convergence speed.
2023	Veerasamy, Veerapandiyan *et al.* [97]	Stability and security	Blockchain	Proposed a PID-based frequency control method and blockchain-based trading for an isolated microgrid. The controller's robustness was validated, considering uncertainty.
2023	Wiesner, Philipp *et al.* [98]	Scheduling and optimizations	Deep neural network	Proposed FedZero, utilizing excess energy from renewable sources, along with underutilized computing resources in data centres, to train machine learning models.
2023	Sharma, Desh Deepak *et al.* [99]	Smart trading	Deep neural network	Addressed blockchain interoperability issues and designed smart contract in microgrid systems, achieving peer-to-peer smart power exchange.

Table 2.9: Federated learning applications in grids, table 3.

Year	Reference	Field	Based on	Contribution
2023	Naidji, Ilyes *et al.* [100]	Stability and security	Deep neural network	Proposed a decentralized, federated learning architecture for networked microgrids, aiming to prevent a single point of failure and maintain energy self-efficiency.
2023	Veerasamy, Veerapandiyan *et al.* [101]	Stability and security	Second-order recurrent neural network	Proposed a federated learning controller for distributed frequency control in multi-microgrid systems, capable of regulating system frequency and maintaining stability even under communication failures and malicious attacks.
2023	Rudayaraj, Andrew Xavier Raj *et al.* [102]	Stability and security	Zeroing neural network	Proposed a federated learning and zeroing neural network-based PID control strategy, effectively enhanced frequency control performance in multi-microgrid systems.
2023	Ren, Chao *et al.* [103]	Stability and security	Four kinds of model	Proposed a secure distributed stability assessment method based on federated learning and differential privacy.
2023	Bondok, Atef *et al.* [104]	Stability and security	Convolutional neural network	Investigated the sensitivity of traditional electricity theft classifiers with federated learning and explored the effectiveness of adversarial training in protecting from electricity theft.
2023	Moayyed, Hamed *et al.* [105]	Stability and security	Convolutional neural network	Proposed a federated learning model for dynamic line rating forecasting, trained on data from nine different regions in Iran.
2023	Xu, Qi *et al.* [106]	Scheduling and optimizations	Transfer Learning	Proposed a federated transfer learning algorithm FTL-EDGE, achieving cost optimization and enhancing the overall economic efficiency of the microgrid.
2023	Wang, Zhenyi *et al.* [107]	Scheduling and optimizations	Multi-agent deep reinforcement learning	Proposed a federated multi-agent deep reinforcement learning algorithm, aiming to minimize costs and achieve self-sufficiency in microgrids.

Federated predictions

In 2020, Zhang, Xiaoning *et al.* proposed a federated two-layer Bayesian LSTM network for predicting solar irradiation [82], using two real-world datasets from SolarGIS and the National Solar Radiation Database. The results indicated that the prediction accuracy was lower than centralized architectures but better than scenarios trained individually. In 2022, Gholizadeh, Nastaran discussed the application of federated learning in short-term load forecasting for power systems [83], comparing it with centralized and local learning schemes. The study demonstrated that federated learning can effectively predict individual and aggregate loads while protecting user privacy.

Other recent research papers also have focused on federated predictions [91] [111] [92] [93] [94], where federated learning was primarily used to protect users' load information and to eliminate the statistical uncertainty in prediction.

Federated scheduling and optimizations

In 2022, Wang, Zhenyi *et al.* introduced a privacy-preserving framework combining federated learning and transfer learning to evaluate the regulation capacity of HVAC (Heating, Ventilation, and Air Conditioning) systems in different types of buildings [86]. Specifically, it proposed a classified federated learning algorithm that allowed each building to train its model locally without sharing data, thus protecting privacy. In 2023, Li, Yuanzheng *et al.* proposed a federated multi-agent deep reinforcement learning algorithm [107], based on physics-informed rewards, aimed at minimizing economic costs and maintaining self-sufficiency in multi-microgrids. The algorithm was trained through a federated learning mechanism, ensuring the privacy and security of each agent's power data.

Other recent research papers have also focused on federated scheduling or optimization algorithms [95] [112] [96] [98]. Federated learning was mainly used to protect the electric power privacy data in microgrids.

Federated stability and security in microgrid

In 2022, Lin, Jun *et al.* introduced a federated learning model named Channel Attention-based Convolutional Neural Network [87], which was used to diagnose fault types in power transformers. The model was updated within the federated learning framework using a hierarchical parameter aggregation strategy to protect data privacy. In 2022, Li, Yang *et al.* presented a false data injection attack detection method based on federated learning and secure deep learning [88], applied to smart grids, with the aim of addressing data privacy and detection performance concerns. The effectiveness of this approach was demonstrated through simulations on IEEE 14 and 118 bus test systems.

Other recent research papers have also focused on federated stability and security in microgrids [101] [102] [103] [104] [105] [99] [100], the microgrid system could determine if it had been attacked using private data while ensuring data privacy protection.

It can be observed that federated learning has been applied in various areas of microgrids in recent years, including transactions, load forecasting, scheduling optimization algorithms, and security protection algorithms. In addition to privacy protection, when combined with edge computing, it can also alleviate the burden on central servers [106] [113] [89] [90].

2.2.4 The attacks and defences in federated learning

Recently, research has found that federated learning cannot fully protect data privacy, as it may be susceptible to various attacks. The thesis primarily focuses on deep leakage attacks [114] (GS attack), as they have the potential to recover original training data and are more destructive compared to other attacks.

In 2019, Zhu, Ligeng *et al.* proposed the Deep Leakage from Gradients (DLG) algorithm [114]. In DLG, the dummy gradients were generated by randomly initialized dummy data and labels. Then, the L2 distance between the dummy gradients and the real gradients from real data was optimized until the dummy data approximated the real data. Simulations found that federated learning cannot fully protect

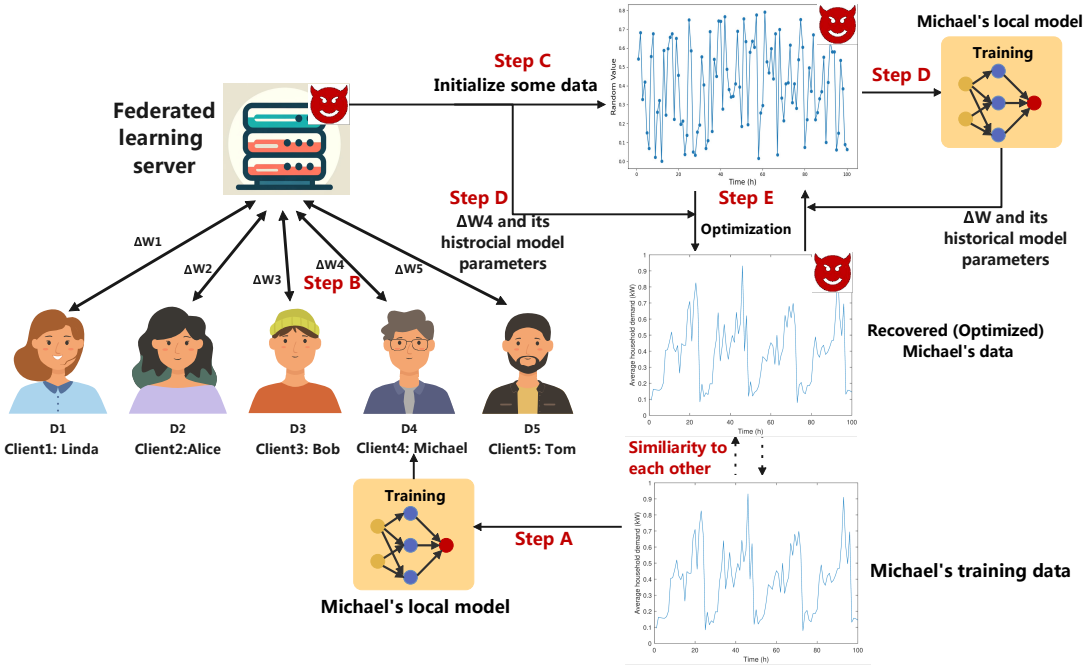

Figure 2.3: A diagram of a GS attack, where Michael's private data is attacked and recovered by a malicious server, thus compromising privacy.

privacy under DLG attacks, as the shared gradients still contain significant private data that can be recovered.

In 2020, Geiping, Jonas *et al.* proposed GS algorithm, which used cosine distance instead of L2 distance in DLG, and regularization terms were added [115]. Additionally, theoretical proofs supporting the proposed algorithm were added, and simulations revealed that its accuracy surpasses that of DLG.

Deep leakage attacks occur in the gap between two model updates in federated learning. Take the attack that happened on the Server as an example. The GS algorithm is shown in Fig. 2.3.

Steps A and B refers to Michael's local training, the inputs can produce specific gradients, which are recorded and donated as 'real gradients' $\nabla L(w)$. In Step C, some dummy data and labels are generated by the malicious hacker on the server. These dummy data are trained and then generated 'dummy gradients' $L(w_{\mathbb{D}})$, also recorded by the attackers in Step D. In Step E, Deep leakage attacks use the L-BFGS optimizer to minimize the distance between real and dummy gradients in (2.4), with

dummy data and labels as optimization variables.

$$\{mb_{\mathbb{D}*}, m_{\mathbb{D}*}\} = \underset{mb_{\mathbb{D}}, m_{\mathbb{D}}}{argmin} \frac{\nabla L(w) \cdot \nabla L(w_{\mathbb{D}})}{\|\nabla L(w)\| \|\nabla L(w_{\mathbb{D}})\|}, \tag{2.4}$$

where superscript $\mathbb{D}$ represents the randomly initialized (dummy). $mb_{\mathbb{D}}$ represents the randomly initialized (dummy) data, $m_{\mathbb{D}}$ represent the randomly initialized (dummy) labels. The model parameters $w_{\mathbb{D}}$ are calculated with $mb_{\mathbb{D}}$ and $m_{\mathbb{D}}$. The $\nabla L(w_{\mathbb{D}})$ are the gradients generated by $mb_{\mathbb{D}}$, $m_{\mathbb{D}}$ and $w_{\mathbb{D}}$. $|| \cdot ||$ refers to the norm.

The dummy data and labels are then optimized to values close to Michael's real data and labels, as is shown in the middle of Fig 2.3 below Step E. Finally, the optimized data similar to the real input data and Michael's privacy is compromised.

Other federated learning attack algorithms from the gradient leakage family are summarized. In 2020, Based on DLG, Zhao, Bo *et al.* proposed iDLG, which introduced a method to recover real labels according to the last fully connected layer [116]. It can effectively improve the accuracy of the DLG algorithm in recovering original data with the help of real label recovery. In 2020, Wei, Wenqi *et al.* proposed client privacy leakage extended iDLG to multi-batch federated learning based on iDLG and introduced prior into the attack system [117]. In 2021, Yin, Hongxu *et al.* proposed the GradInversion algorithm to recover multiple labels of batch data in the image classification problem [118]. The algorithm improved attack efficiency.

Please note that sometimes the training data in federated learning may contain random noise from telecommunications, or the noise may be intentionally added (e.g., using differential privacy algorithms) to protect the privacy of the original data. For the DLG family of algorithms, if noise is present, the original data with the added noise should be considered as a whole, and the restored results after using the DLG algorithm will also have the original data with the noise. In other words, the DLG algorithms cannot act as a filter to eliminate noise. This thesis assumes that the received data has undergone noise reduction processing, as described in Chapter 3, section 3.1.2 'Format conversion and noise filtering'.

The attack mentioned above algorithms are frequently simulated and tested in natural language processing and computer vision; however, they have yet to be

considered in electricity areas, such as smart homes or microgrid scenarios. Chapter 5 proposes a framework for measuring federated learning privacy leakage based on the GS algorithm.

2.3 Artificial intelligence and machine learning

AI belongs to the computer science area and aims to mimic human intelligence by developing intelligent algorithms. Machine learning is a critical pathway to achieving the goals of artificial intelligence, involving designing algorithms and models that enable computers to learn from data and make decisions or predictions [122] [123]. Machine learning methods can be categorized into three main types based on the nature of the data and the learning process.

1. Supervised learning uses labelled datasets to train algorithms, allowing models to make accurate predictions or classifications, and excels in areas like computer vision and natural language processing.

2. Unsupervised learning deals with unlabeled data, often used in cluster analysis and anomaly detection, aiming to uncover hidden patterns and structures.

3. Reinforcement learning guides the learning process through interaction with the environment and received rewards, having extensive applications in gaming and robot navigation.

In this section, the main focus is on deep learning and reinforcement learning. In the deep learning part, the emphasis is on neural networks, embedding, and knowledge graphs. In the reinforcement learning part, the emphasis is on deep reinforcement learning algorithms.

2.3.1 Deep learning

With the digitalization trends, there is continuous innovation in data, algorithms, and hardware. Deep learning technology has achieved significant success in recent years. For example, high-precision expert question-answer systems, large models,

Table 2.10: Attack algorithms in federated learning.

Year	Reference	Field	Name	Based on	Contribution
2015	Fredrikson, Matt *et al.* [119]	Model inversion attack	Gradient descent	Optimization	Proposed a method in which attackers define and use a cost function, and retrived the private training data based on minimizing that cost function.
2017	Shokri, Reza *et al.* [120]	Memebership inference attack	Memebership inference attack	Training data and test data have different entropy.	Proposed an algorithm to infer whether a data is part of the model's training data.
2017	Hitaj, Briland *et al.* [121]	GAN-based attack	Deep models under the GAN	Generative adversarial network	Proposed an attack algorithm creating data that had a similar distribution with the private training data with Generative adversarial networks.
2019	Zhu, Ligeng *et al.* [114]	Deep leakage attacks	Deep leakage from gradients	L-BFGS optimization	Proposed an algorithm in recovering the training data by optimizing the real gradient generated by clients and dummy data generated by attackers.
2020	Geiping, Jonas *et al.* [115]	Deep leakage attacks	Gradient similarity	Deep leakage from gradients	Based on DLG, using cosine similarity instead of L2 distance, and regularization terms were added.
2020	Zhao, Bo *et al.* [116]	Deep leakage attacks	Improved deep leakage from gradients	Deep leakage from gradients	Proposed a method to recover real data labels according to the last fully connected layer of the model on image classification problems.
2020	Wei, Wenqi *et al.* [117]	Deep leakage attacks	Client Privacy leakage	Deep leakage from gradients	Proposed a federated learning framework, recovering real data labels according to the last fully connected layer of the model.
2021	Yin, Hongxu *et al.* [118]	Deep leakage attacks	GradInversion	Client Privacy leakage algorithm	Recovered multiple labels of batch data in the image classification problem.

and the ChatGPT algorithm have been developed in the natural language processing domain based on Transformer architectures [124] [125]. Image processing, object detection, face recognition, and other tasks have been developed in the computer vision area by the toolboxes provided by OpenCV [126]. Combining users' viewing histories and search behaviours, personalized recommendation algorithms have been developed by Amazon and YouTube, effectively improving their user retention rates [127] [128]. All these cases have highlighted that AI algorithms have deeply penetrated and impacted human life.

Deep learning's essence lies in neural networks, which consist of various layers, including an input layer, hidden layers, and an output layer, each comprising numerous interconnected neurons or nodes [129]. In deep learning, one of the key techniques is the backpropagation algorithm, which uses gradient descent to update network weights, thereby training the model [130]. This process demands a high volume of data and computational power, accordingly, as these two aspects grow, deep learning performance also improves.

Common deep learning architectures include DNN for classification and regression problems, CNN for image processing tasks [131], RNN and LSTM for time series analysis and language modelling tasks [132] [133], Generative Adversarial Network (GAN) for data-generating tasks, Transformer for natural language processing [134], Graph Neural Network (GNN) for graph data and knowledge graphs problems [135], etc. CNN, RNN, LSTM and GNN are used in chapter 3-5, and they are introduced in formula 2.5 to 2.14.

Deep Neural Network

Many papers and authors influenced the development of DNN, which consisted of multiple fully connected layers composed of many neurons (nodes). These layers were arranged in sequence, where the output of one layer served as the input for the next layer. DNNs formed the foundation of other network structures.

For the a-th layer of the DNN and this layer's input y_{a-1}, (2.5) describes the fully connected layer and its output $(W_a y_{a-1} + b_a)$, which then gives this a-th layer

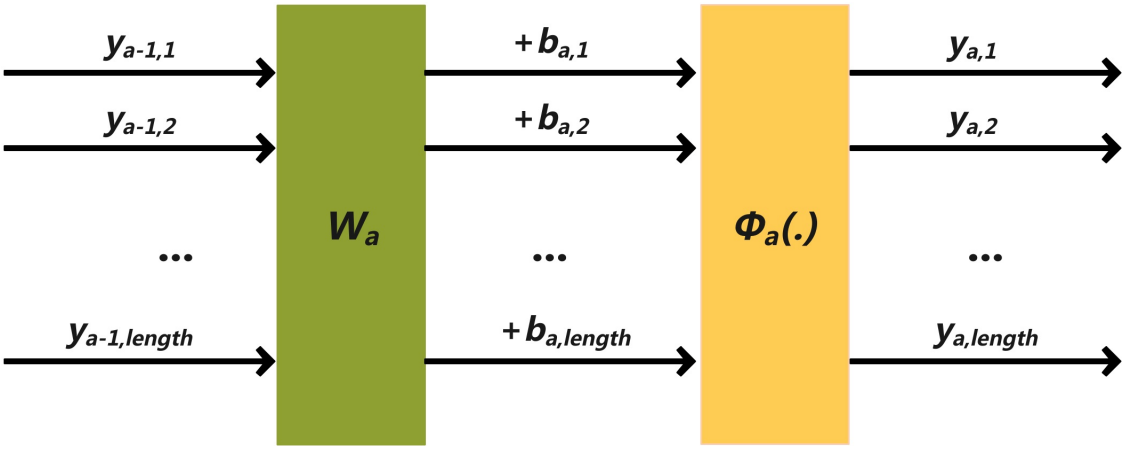

Figure 2.4: The structure of DNN.

output y_a after the activation function $\Phi_a(\cdot)$ [129].

$$y_a = \Phi_a(W_a y_{a-1} + b_a), \tag{2.5}$$

where Φ is the activate functions such as sigmoid or relu. W_a is the weights of the layer a, accordingly, b_a is the bias of the layer a. The structure of DNN is shown in Fig. 2.4, *length* is the length of the elements in y and b.

Convolutional Neural Network

In 2014, Kim, Yoon *et al.* proposed CNN [131], a type of deep learning model designed for processing image data with a grid-like structure. The core feature of CNNs was the use of convolutional layers, which extracted features from the input data through convolution operations using filters called kernels. For a CNN, which is particularly effective for spatial data processing like images, the convolution operation for a layer can be expressed as [131]

$$y_a = \Phi_a\left(\sum_{row}\sum_{col} W_{a,row,col} y_{a-1,i+row,j+col} + b_a\right), \tag{2.6}$$

where y_a represents the output of layer a, Φ is the non-linear activation function, $W_{a,row,col}$ is the weight of the convolution kernel, $y_{a-1,i+row,j+col}$ refers to the elements of the input feature map that the kernel covers, where i and j are the coordinates of the output y_a, *row* and *col* are the coordinates of the kernel W_a, b_a is the bias

term.

Recurrent Neural Network

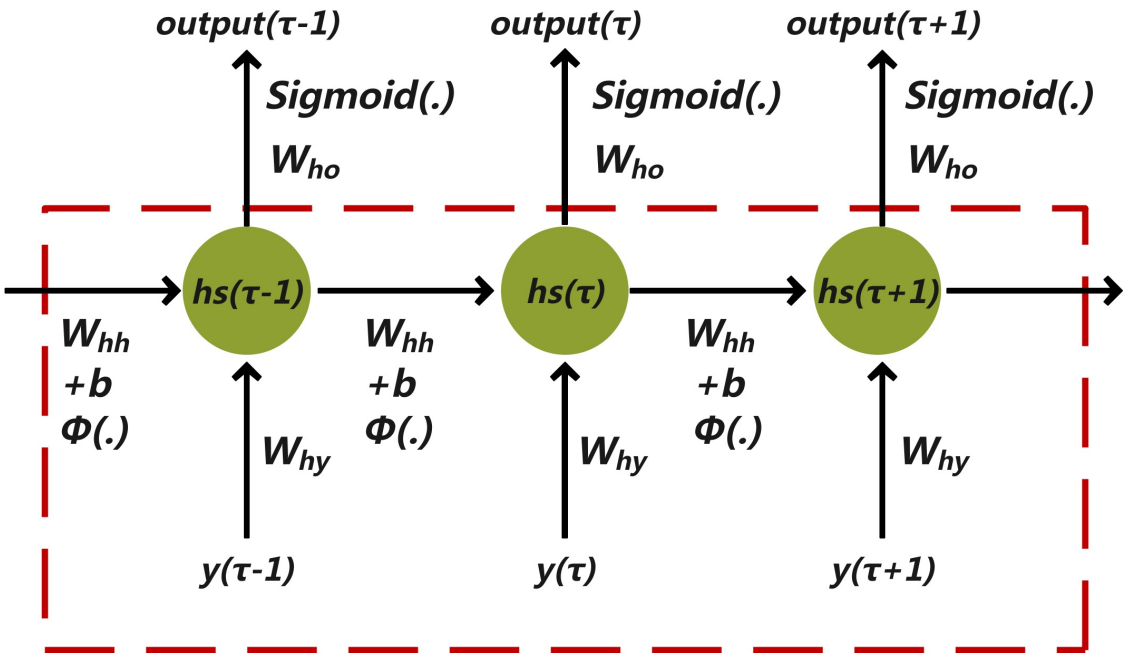

Figure 2.5: The structure of RNN, the area enclosed by the red dashed line is the range described by the formula (2.7).

In 1990, Elman, Jeffrey L *et al.* proposed RNN [132], a type of neural network specifically designed for processing sequential data, making them well-suited for time series or natural language datasets. RNNs processed each sequence element by updating this internal state, which contained information from previous steps, allowing the network to capture temporal dependencies. For an RNN, considering it processes data sequentially where the output at each time step depends on the current input and the previous state, the equation can be represented as [136]

$$hs(\tau) = \Phi\left(W_{hh}hs(\tau-1) + W_{hy}y(\tau) + b\right), \tag{2.7}$$

where $hs(\tau)$ represents the output hidden state at time step τ, Φ is the activation function, normally tanh is used. W_{hh} is the weight matrix for the transition from the previous output (or hidden state) $hs(\tau-1)$ to the current hidden state, W_{hy} is the weight matrix for the transition from the current input to the hidden state, $y(\tau)$

is the input. b is the bias term.

The structure of RNN is shown in Fig. 2.5, where the area enclosed by the red dashed line is the range described by the formula (2.7), the *output* of RNN can be gotten from the hs by using fully connected layers with weights W_{ho} and activation function Sigmoid.

Long Short Term Memory

In 1997, Hochreiter, Sepp *et al.* proposed LSTM, a special RNN designed to handle sequence data with long-term dependencies [133]. The uniqueness of LSTMs was their internal structure, which included several gates controlling the flow of information, thus addressing the vanishing or exploding gradients that traditional RNNs face while processing long sequences. A standard LSTM can be given from (2.8) to (2.14) [137].

$$ai_v = \text{sigmoid}(W_{\text{in}} \cdot [hs_{v-1}, u_v] + b_{\text{in}}), \tag{2.8}$$

$$fg_v = \text{sigmoid}(W_{fg} \cdot [hs_{v-1}, u_v] + b_{fg}), \tag{2.9}$$

$$gc_v = \tanh(W_{gc} \cdot [hs_{v-1}, u_v] + b_{gc}), \tag{2.10}$$

$$o_v = \text{sigmoid}(W_o \cdot [hs_{v-1}, u_v] + b_o), \tag{2.11}$$

$$cs_v = cs_{v-1} \cdot fg_v + ai_v \cdot gc_v \ , \tag{2.12}$$

$$hs_v = \tanh(cs_v) \cdot o_v \ , \tag{2.13}$$

$$U_v = \text{regression}(W_v \cdot hs_v + b_U), \tag{2.14}$$

where ai_v represents the activate value, and sigmoid and tanh are the activation functions. The W represents weight, with the subscripts in representing the input gate, and gc representing the weight or bias of the cell candidate, the subscript fg representing the forgotten gate, and the subscript o representing the output gate. hs_v is the hidden state, u_v is the input layer, v = 1,2,3,... v', v' is the sequence length. b is the bias of 3 gates, b_U is the bias of full connection layer. fg_v represents the forgotten gate, gc_v represent the input gate call candidate, o_v represents the output gate. cs_v is the cell state. The overall structure of LSTM and equations

from (2.8) to (2.13) can be presented by Fig. 2.6.

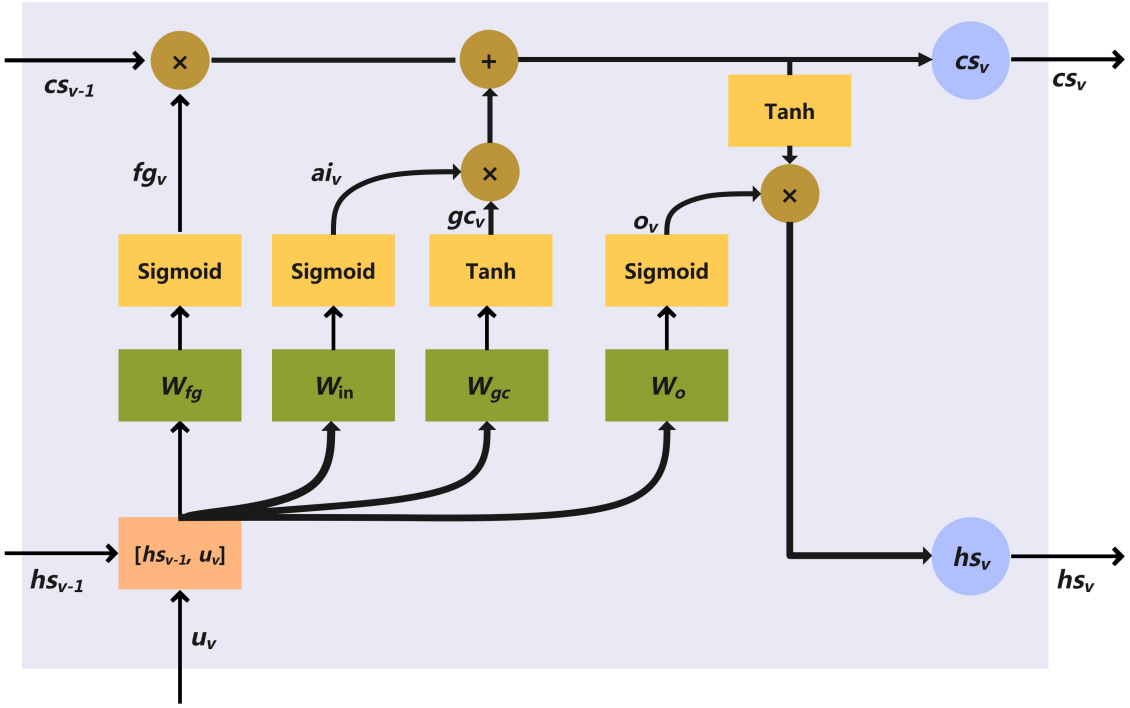

Figure 2.6: The structure of LSTM, including input, forgotten, output gates.

Graph Neural Networks

In GNNs, data was represented as a graph, where nodes represented entities and edges represented relationships between entities. The key to GNNs was the method of neighbourhood aggregation, where each node updated its representation by aggregating information from its neighbouring nodes. Common types of graph neural networks included Graph convolutional network [138], Graph attention network [140] and Graph Sample and Aggregate (GraphSAGE) [139].

Deep learning algorithms provide powerful tools for solving complex problems, table 2.11 summarizes common deep learning algorithms. In the future, it will continue to drive technological innovation across various industries.

2.3.2 Recommendation systems

As human society progresses, the amount of information has grown exponentially, making it difficult for individuals to filter through the vast data and find relevant

Table 2.11: Literature review on deep learning algorithms.

Year	Reference	Field	Name	Based on	Contribution
1986	Rumelhart, David E *et al.* [136]	Deep learning	Deep neural network	Backpropagation algorithm	A powerful machine learning model with multiple layers, used for complex data analysis and pattern recognition.
1990	Elman, Jeffrey L *et al.* [132]	Deep learning	Recurrent neural networks	Deep neural network	Designed for sequential data, with natural language processing and time series analysis applications.
1997	Hochreiter, Sepp *et al.* [133]	Deep learning	Long short-term memory	Deep neural network	A specialized recurrent neural network architecture designed for handling long-range dependencies in sequential data.
2014	Kim, Yoon *et al.* [131]	Deep learning	Convolutional neural network	Deep neural network	A deep learning model for image processing and pattern recognition, used in computer vision tasks.
2016	Kipf, Thomas N *et al.* [138]	Deep learning on graph	Graph convolutional network	Graph neural network	A graph neural network model for graph-structured data, used in graph analytics and node classification tasks.
2017	Vaswani, Ashish *et al.* [124]	Deep learning	Transformer	Attention mechanism	A neural network architecture for sequence-to-sequence tasks, known for its self-attention mechanism and applications in natural language processing.
2017	Hamilton, Will *et al.* [139]	Deep learning on graph	GraphSAGE	Graph neural network	A method for sampling neighboring nodes and aggregating their information, made it suitable for handling large-scale graph data.
2017	Veličković *et al.* [140]	Deep learning on graph	Graph attention network	Graph neural network	A neural network model for graph data, used attention mechanisms for node aggregations.

information. Recommendation systems have emerged to address the problem of information overload [141]. The recommendation system is a predictive sorting problem that helps users select a practical list of items from a massive dataset. Collaborative filtering and factorization machines are traditional recommendation algorithms [142] [143]. AI algorithms have been applied in recommendation systems, such as AutoRec, Wide & Deep, Graph neural networks, etc. [144]. This thesis focuses on AI-based graph recommendation systems, including embeddings and knowledge graph embedding algorithms. In applying recommendation algorithms, the embedding layer is often needed and placed before the input layer for data mapping purposes, or embedding can be used as recommendations.

Embedding

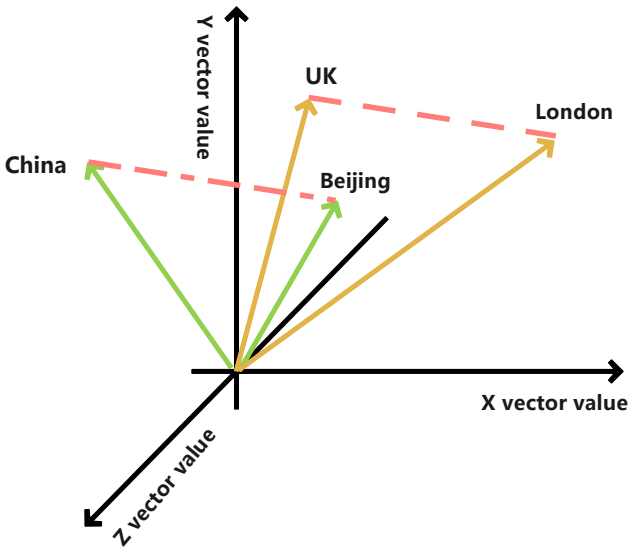

Figure 2.7: A diagram of trained word embeddings in a three-dimensional space. After training, Embedding (UK) - Embedding (London) is approximately equal to Embedding (China) - Embedding (Beijing).

Embedding is a widely used technique in recommendation systems that transforms sparse vectors (objects) into a low-dimensional dense vector. This object could be a word, a product, or other entities. Embedding can essentially encode any information and, once trained, contains a wealth of valuable insights into information. Like neural networks, embedding vectors can learn to express features of the corresponding objects through backpropagation, with the distance between vectors re-

flecting the similarity between objects. Fundamental embedding algorithms include Word2Vec [145] and Item2Vec [146]. Embedding vectors can also be manipulated mathematically; for instance, after training with the Word2Vec algorithm, Embedding (UK) - Embedding (London) is approximately equal to Embedding (China) - Embedding (Beijing), as is shown in the Fig. 2.7. This indicates that embeddings can unearth latent relationships, such as the capital-country relationship.

Knowledge graph

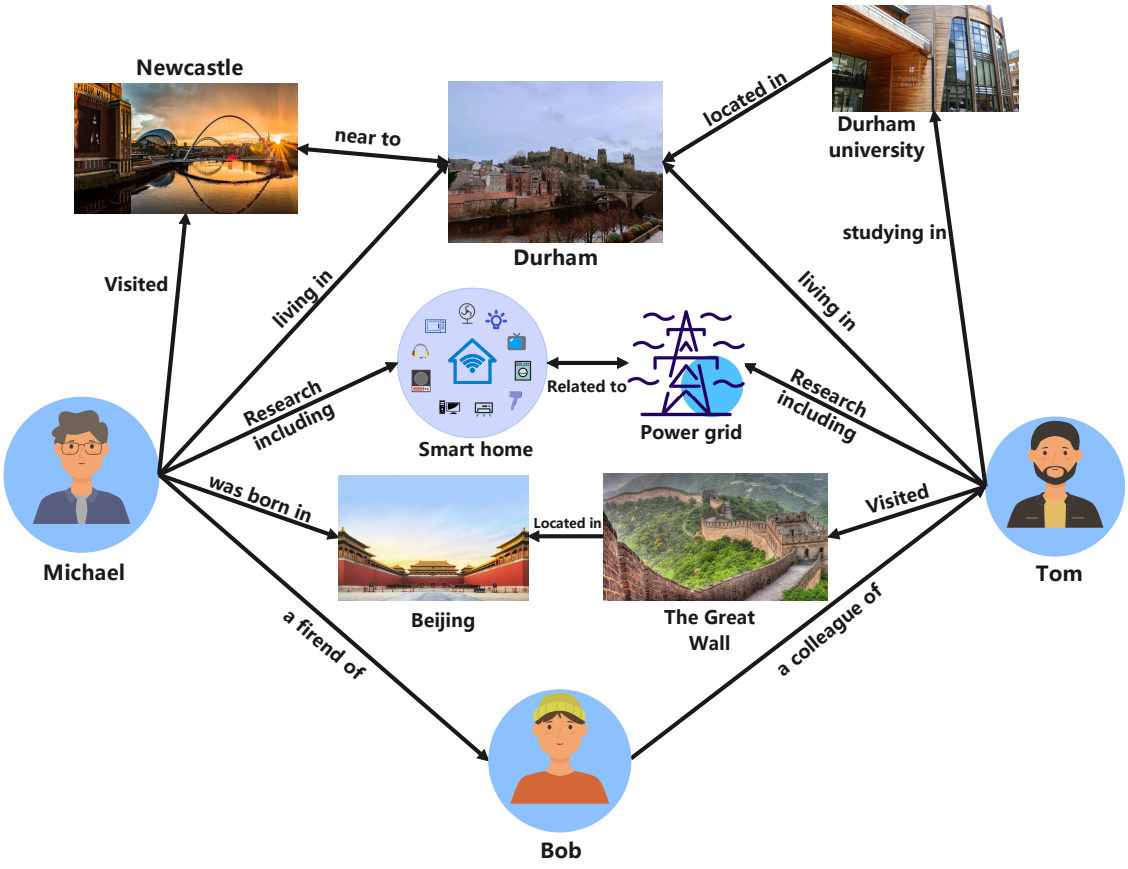

Figure 2.8: Schematic diagram of a knowledge graph, depicting a recommendation system where Tom is recommended to Michael for four different reasons, which can be understood through the connections in the graph.

The data in recommendation systems is often irregular, and such data can be represented with graph structures, facilitating the incorporation of common sense and making recommendations effective. Fig. 2.8 shows how the recommendation system recommends Tom to Michael for different reasons, including living in Durham, similar research directions, similar visiting experiences and common friends.

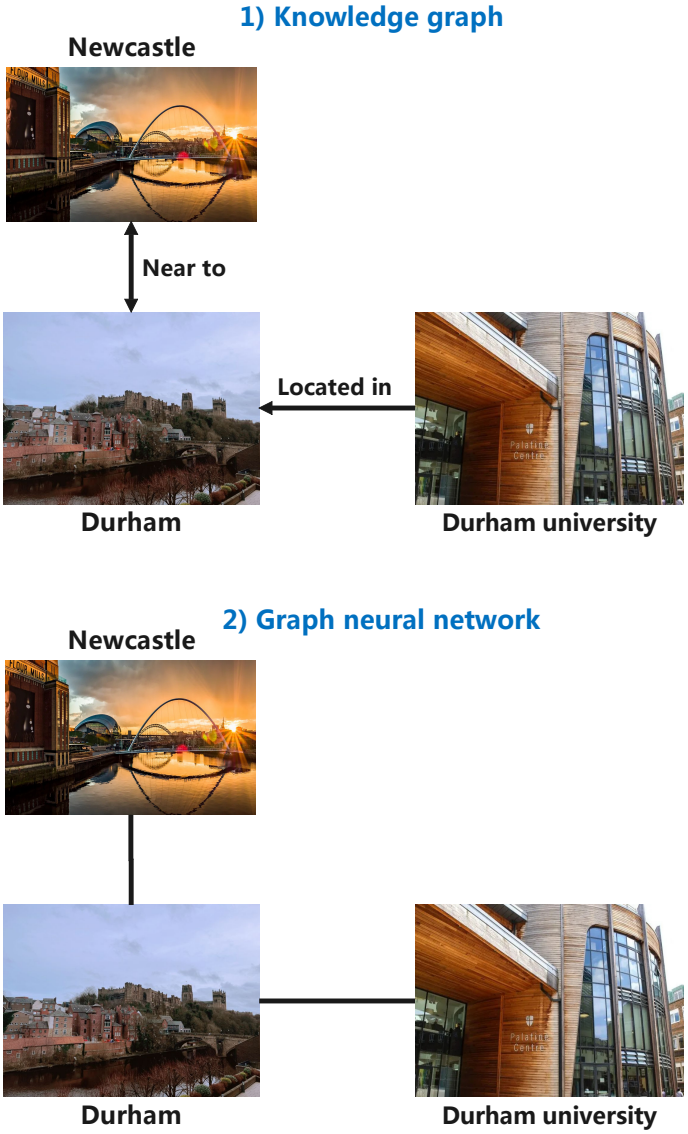

Figure 2.9: Compare their differences: 1) Knowledge Graph and 2) Graph Neural Network. In knowledge graphs, the attributes of the edges cannot be ignored.

There are two kinds of graphs in algorithms: knowledge graph and graph neural network algorithm. Fig. 2.9 shows the difference between graph neural networks and knowledge graphs. In graph neural networks, the edge information is ignored, while in knowledge graphs, the graph edges' characteristics are considered, making them aligned with real-world applications. The knowledge graph is the focus of this thesis, and it provides the rules of graph embedding aggregations.

Knowledge graph embedding algorithm

When dealing with graph data, graph embedding is used instead of normal embeddings. Graph embedding is designed to represent graph-structured data. In a graph, nodes typically represent head or tail entities, and edges represent the relationships between these entities. Graph embedding aims to map nodes and edges into a low-dimensional vector space through learning to preserve the graph's structural information better. After the embedding layer, the embeddings of data can be sent to the knowledge graph layer for recommendations.

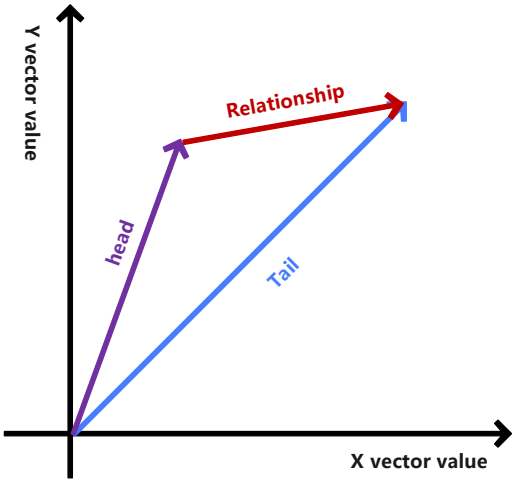

Figure 2.10: An illustration of TransE in a two-dimensional space, where the head embedding plus relation embedding approach tail embedding after training.

In 2013, Bordes *et al.* proposed Translating Embeddings for Modeling Multi-relational Data (TransE), a typical knowledge graph embedding algorithm [147], as shown in Fig. 2.10. It represented entities and relations as low-dimensional vectors, such that the vector of the head entity plus a relation vector approximated the tail entity vector. This method was particularly suited for handling one-to-one relationships in large-scale knowledge graphs, its formula is expressed as

$$TransE_t \approx TransE_h + TransE_r \ , \tag{2.15}$$

where $TransE_t$ represents the embedding of the tail node in the knowledge graph. Correspondingly, $TransE_h$ represents the embedding of the head entity (the head node in the knowledge graph), and $TransE_r$ represents the embedding of the relation

(the relation on the edge in the knowledge graph).

In addition to TransE, in 2014, Perozzi, Bryan proposed DeepWalk, a graph embedding algorithm that generated sequences of nodes by performing random walks on a graph [148]. The advantage of DeepWalk was its effectiveness in capturing the neighbourhood relationships between nodes in a graph.

In 2016, Grover Aditya proposed Node2Vec, explored nodes with a random walk algorithm, and trained embeddings with Word2Vec [149]. This algorithm balanced the exploration of local neighbourhoods and relationships with more distant nodes.

The knowledge graph convolutional network algorithm (KGCN) and KGAT algorithm are other well-known graph embedding algorithms [135] [150]. Both of them combine knowledge graphs with graph embeddings, and they are commonly used in recommendations. In 2019, Wang, Xiang *et al.* proposed the KGAT algorithm, which utilized graph attention networks for information passing between the head, relation and tail embeddings, aggregating them, and making recommendations [150]. In KGAT, attention mechanisms are utilized to determine the aggregation weights of neighbouring node embeddings in the formula (2.16); the schematic diagram of the embedding aggregation in the KGAT algorithm is shown in Fig. 2.11 [150].

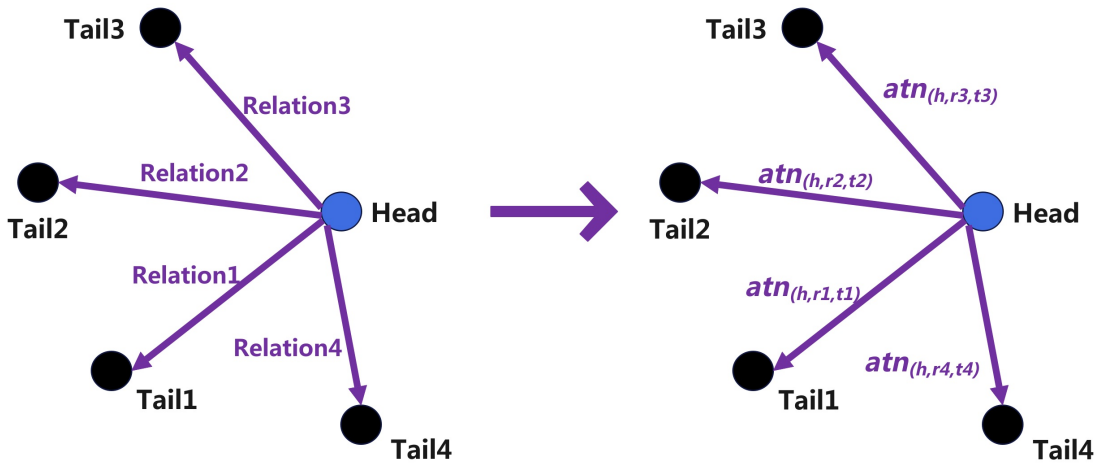

Figure 2.11: The schematic diagram of the embedding aggregation part of the KGAT. The attention atn is calculated for embedding aggregations.

$$atn_{(h,r,t)} = \text{Softmax}((W_r em_t)^{\mathrm{T}} \tanh(W_r em_h + em_r)) \ , \tag{2.16}$$

where the $atn_{(h,r,t)}$ is the calculated attention for knowledge graph aggregation.

Softmax and tanh (hyperbolic tangent function) are activation functions. W_r is the relation weight matrix. em_h, em_r and em_t are the head, relation and tail entity respectively. Then, it aggregates the embeddings of neighbouring nodes based on the weights in (2.17) [150]

$$em_{Nh} = \sum_{(h,r,t)\in Nh} atn_{(h,r,t)} \times em_t \ , \tag{2.17}$$

where em_{Nh} is the aggregated entity, which means how much information that the tail entity em_t is going to pass to the head entity em_h, which is represented in (2.18) [150]

$$agg = \text{LeakyRelu}(W_{trans}(em_h + em_{Nh})) \ , \tag{2.18}$$

where agg is the information aggregation vector, which contains weighted information with the head and tails entities, and can then be used for recommendations by embedding dot products with the aimed user embeddings. LeakyRelu is the activation function. W_{trans} is the transformation matrix.

Other well-known knowledge graph-based algorithms include Graph convolutional networks (proposed by Schlichtkrull Michael *et al.*, 2018) [151], Knowledge graph convolutional networks (proposed by Wang *et al.*, 2019) [135], Knowledge graph neural network with Label Smoothing (proposed by Wang, Hongwei *et al.*, 2019) [135], and Knowledge graph factorization machine (proposed by Anelli *et al.*, 2019) [152]. All the related algorithms in this sub-section are summarized in table 2.12.

2.3.3 Deep reinforcement learning

Although deep learning has capabilities in data classification, prediction, and sorting problems, it is not enough to complete an intelligent system. Similar to humans, for the intelligent system, not only is learning from given data needed, but it also needs to learn interactions with the real world. Reinforcement learning models the natural environment into several parts: environment, agent, action, state and reward [156] [157]. The agent interacts with the environment through actions and receives

Table 2.12: Literature review on recommendation algorithms.

Year	Reference	Field	Name	Based on	Contribution
2013	Mikolov, Tomas *et al.* [153]	Embedding	Word2Vec	Deep neural network	Mapped vocabulary to fixed-size dense vectors by learning from text data.
2013	Bordes, Antoine *et al.* [147]	Graph embedding	TransE	Graph neural network	Embedded entities and relations in a knowledge graph by modelling translations in vector space.
2014	Perozzi, Bryan *et al.* [148]	Graph embedding	Deepwalk	Graph theory	Generated graph embeddings, using random walks to capture network structural information.
2016	Grover, Aditya *et al.* [149]	Graph embedding	Node2Vec	Word2Vec	Embedded nodes in a graph by balancing breadth-first and depth-first random walks.
2016	Joulin, Armand *et al.* [145]	Embedding	FastText	Word2Vec	Word embedding algorithm that created document vectors by stacking and averaging the vectors of words.
2017	He, Xiangnan *et al.* [154]	Recommendation system	Neural collaborative filtering	Collaborative filtering	Collaborated filtering algorithm with neural networks for personalized recommendations.
2018	Devlin, Jacob *et al.* [155]	Embedding	BERT	Transformer	Used the bidirectional encoder of the Transformer algorithm to generate contextually relevant word embeddings.
2019	Wang, Xiang *et al.* [150]	Graph embedding	Knowledge graph attention network	Graph attention network	Used attention mechanisms for knowledge graph propagation and transformation.
2019	Wang, Hongwei *et al.* [135]	Graph embedding	Knowledge graph convolutional network	Graph convolutional network	Used matrix-based transformation operations for knowledge graph propagation.

feedback from it. The feedback can be positive (reward) or negative (punishment). The agent needs to interact with the environment, learning a policy to maximize rewards through training. As illustrated in Fig. 2.12, in reinforcement learning, the learning process establishes an interaction loop between the environment and the agent, enabling the agent to learn from experience, become more competent and make wise decisions.

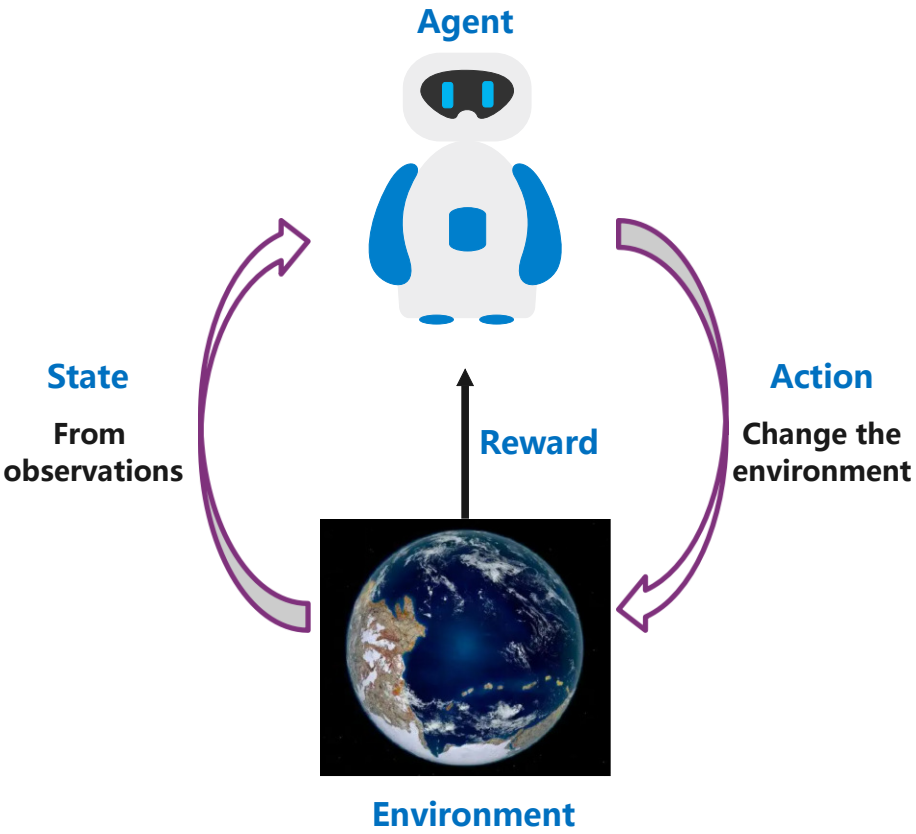

Figure 2.12: Schematic diagram of the reinforcement learning structure, including an agent, environment, actions, rewards, and states.

When dealing with high-dimensional data, traditional reinforcement learning may consume a large amount of resources. In such cases, neural networks can approximate the value function of reinforcement learning. This leads to integrating deep learning and reinforcement learning, resulting in deep reinforcement learning [158] [29]. Deep reinforcement learning can be divided into value-based, policy-based and mixed models.

Value-based methods

This category needs optimizing based on the action-value function, which represents the expected reward of taking a specific action in a given state.

DQN is a typical value-based deep reinforcement learning algorithm that identifies the discrete policy that maximizes the Q value. In 2015, Volodymyr, Mnih *et al.* proposed Deep Q-Network [137]. DQN algorithm approximated the Q-function using deep neural networks and introduced experience replay to disrupt the temporal correlation of samples. It achieved professional levels in the field of gaming decisions. DQN is based on the action-value function form of the Bellman equation, which is expressed in [137]

$$\mathbb{B}_{\text{out}} = (rd + \gamma \times Q_{\max_{n+1}}(\mathbb{S}, \mathbb{A}, W_{\text{DQN}})), \tag{2.19}$$

where $\mathbb{B}_{\text{out}}$ is the output of the Bellman equation. DQN uses a neural network to fit the Q-table and to get the Q-value. rd is the instant reward, γ is the discount factor of DQN, $Q_{\max}$ is the maximum Q-value in the future step $n+1$ considering all the possible situations. $\mathbb{S}$ and $\mathbb{A}$ are the state and action collections. W_{DQN} represents the parameters of the DQN.

The training process of Deep Q-Networks involves several key steps, including:

- Initialization: Including the replay memory $\mathbb{M}_r$ to store the agent's experiences, target network $\text{DQN}_{\text{target}}$, online network $\text{DQN}_{\text{online}}$, turns for updating the online network with the target network Tr_{gap} and turns for train the target network with the stored memory Tr_{renew}.

- Exploration: The agent selects an action based on an ϵ-greedy policy and online network $\text{DQN}_{\text{online}}$. After acting, the agent observes the reward and the next state. This transition (current state, action, reward, next state) is stored in the replay memory $\mathbb{M}_r$.

- Training: At regular intervals Tr_{renew}, the algorithm samples a mini-batch of transitions from the replay memory for training the online network $\text{DQN}_{\text{online}}$ with Bellman equation (2.19). At regular intervals Tr_{renew}, the target network

DQN_{target} is updated from online network DQN_{online} with weights.

- Repeat: Repeats for a large number of episodes, gradually improving the policy as the Q-network learns to estimate the action-value function more accurately.

The primary innovations in DQN are experience replay and the target network. The former helps to break the correlation between state transitions, while the latter helps to reduce estimation bias. Both of them enhance the stability of learning.

Based on DQN, in 2016, Wang, Ziyu *et al.* proposed duling DQN [159]. Based on the traditional DQN, Dueling DQN split a Q function into two parts: estimating state value (Value) and each action's advantage (Advantage). This allowed for more effective learning of state values and action advantages. In 2016, Van Hasselt *et al.* proposed double DQN [160]. Based on traditional DQN, double DQN maintained two similar networks: the current network (used for selecting actions) and the target network (used for evaluating actions). Double DQN effectively reduced the overestimation of the value of certain state-action pairs.

Policy-based methods

Policy-based methods learn the mapping from states to actions. In 2015 and 2017, Schulman, John *et al.* proposed trust region policy optimization [161] and proximal policy optimization [162], which aimed to address one of the key challenges in policy gradient methods: how to effectively update policies without sacrificing stability. Trust region policy optimization established a 'trust region' by limiting the KL divergence between the old and the new policies, avoiding significant fluctuations in performance. Proximal policy optimization adjusted the size of its policy updates to keep a reasonable range of the KL divergence change. Both algorithms enhanced the stability.

Mixed algorithms

Mixed algorithms, which combine policy-based optimization with value-based optimization, merge the advantages of the two approaches.

Deep Deterministic Policy Gradient (DDPG) is a typical mixed Actor-Critic algorithm. In 2016, Lillicrap, Timothy P *et al.* proposed DDPG [160]. DDPG combined the experience replay of Q-learning, the target networks, and the characteristics of policy gradient methods, making it particularly suitable for problems with continuous action spaces. Unlike traditional policy gradient methods (which use stochastic policies), DDPG employed a deterministic policy. In DDPG, the Actor outputs a deterministic action, which, along with the state, enters the Critic network (the Q network). The Actor aims to obtain higher scores from the Critic, and the Critic seeks greater returns, similar to achieving higher Q values as in DQN. In 2017, Lowe, Ryan *et al.* proposed MA-DDPG, a multi-agent deep reinforcement learning algorithm that extended the single DDPG algorithm to a multi-agent environment [163]. In MA-DDPG, each agent learned a policy based on a centralized critic network, which considered the observations and actions of all agents while depending on their actor network to generate actions.

In summary, by combining the feature extraction capabilities of deep learning with the decision-making learning mechanisms of reinforcement learning, deep reinforcement learning can achieve significant results in more complex and high-dimensional problems, such as gaming and complex controlling systems. All the related algorithms are summarized in table 2.13.

2.4 Explainable and interpretable machine learning algorithms

The concept of explainable and interpretable machine learning are introduced in this section. They don't belong to any of the categories of machine learning mentioned earlier but focuses on enhancing the transparency and explainability of machine learning models. Understanding and explaining the decisions made by machine learning models is a critical research area, especially in applications where the validation and in-depth analysis of model predictions are needed. Despite the impressive performance of machine learning methods in various tasks, they are often considered 'black boxes', making it challenging to comprehend how they arrive at specific

Table 2.13: Literature review on deep reinforcement learning algorithms.

Year	Reference	Field	Name	Based on	Contribution
2015	Schulman, John *et al.* [161]	Policy based	Trust region policy optimization	General policy gradient	Ensured stable policy updates by constraining the changes to policy distributions within trusted regions.
2015	Volodymyr, Mnih *et al.* [137]	Value based	Deep Q-network	Q-leaning	Combined Q-learning with deep neural networks for reinforcement learning in high-dimensional environments.
2016	Lillicrap, Timothy P *et al.* [164]	Mixed	Deep deterministic policy gradient	Deep Q-leaning	Combined policy gradients with Q-learning for continuous action spaces in reinforcement learning.
2016	Wang, Ziyu *et al.* [159]	Value based	Duling deep Q-network	Deep Q-Network	Enhanced standard DQN by separately estimating state values and the advantage for each action.
2016	Van Hasselt *et al.* [160]	Value based	Doule deep Q-network	Deep Q-Network	Mitigated overestimation in standard DQN, using two networks for more accurate action value estimation.
2017	Schulman, John *et al.* [162]	Policy based	Proximal policy optimization	Trust Region Policy Optimization	Solved the problem of learning new strategies while avoiding performance fluctuations or collapses due to overly large policy updates.
2017	Lowe, Ryan *et al.* [163]	Mixed	Multi agent DDPG	DDPG	Extended DDPG to multi-agent environments.

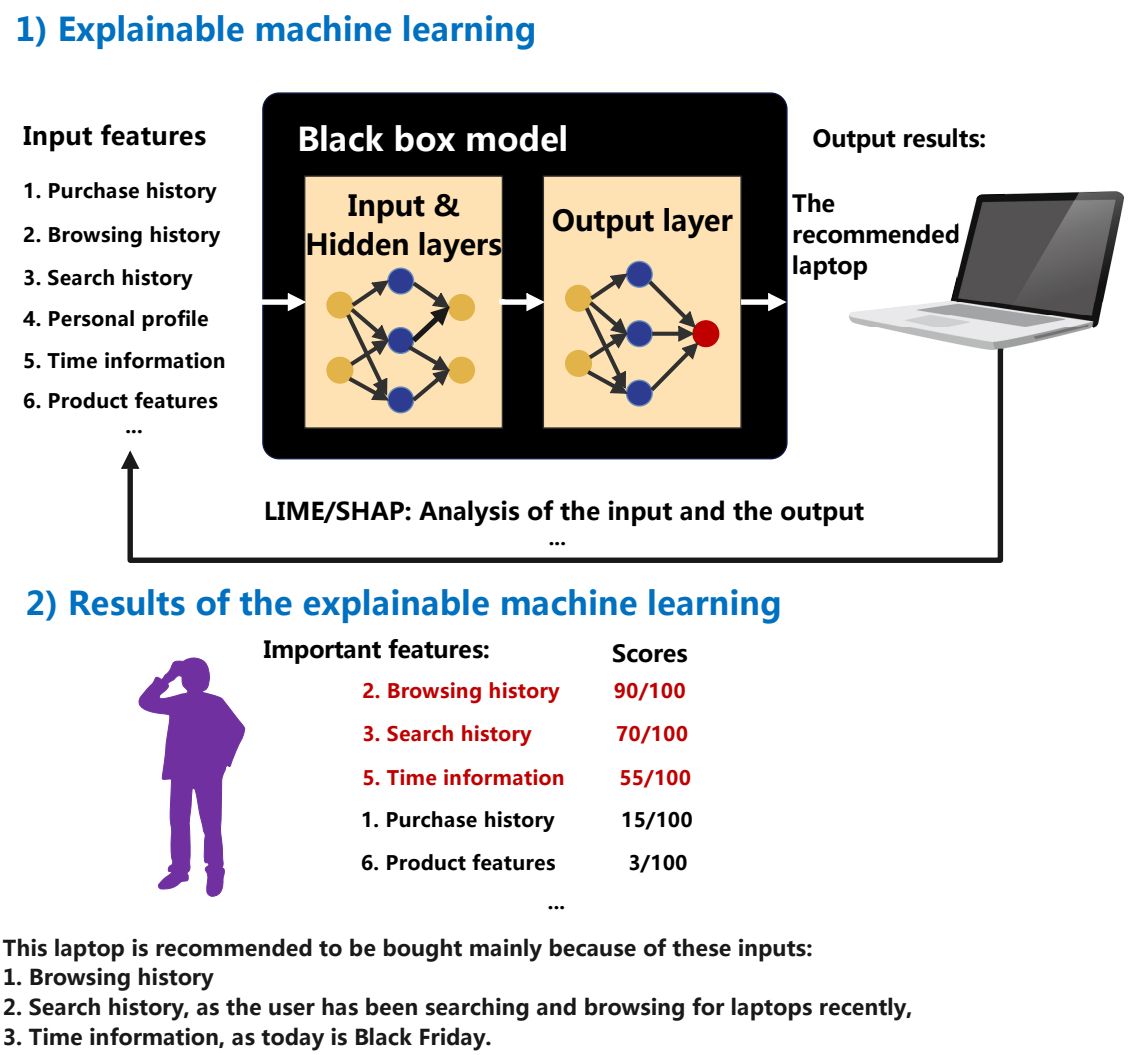

Figure 2.13: Schematic diagram of explainable machine learning: 1) In a recommendation system, the input is 'Input features', and the recommendation result is a Laptop. Using explainable machine learning algorithms to explain it. 2) Schematic diagram of the explainable machine learning results, identifying all the important input features and providing importance scores.

decisions. Explainability is crucial in machine learning models, as it increases the social acceptance of AI. Generally, the more a machine learning decision is related to human life, the more critical it is to explain the machine's behaviour. Explainable machine learning aims to transform machine learning models into logical relationships and rules understandable to humans [165].

Based on the algorithm, explainability can be classified as interpretable or explainable. 'Interpretable' means the model's structure is inherently explainable, such as simple neural networks like linear regression, decision trees or reasoning al-

gorithms in graph neural networks, where each step in the tree or graph can be used to explain and trace the results, as is shown in Fig. 2.8. 'Explainable' refers to enhancing a model's explainability using algorithms after the model has been trained. Fig. 2.13 describes the general usage of explainable machine learning, including 1) A recommendation system recommends a laptop to the user based on input features. 2) An explainable machine learning algorithm is used, which ranks the importance of each input feature based on the input and output values.

Explainable machine learning

Local Interpretable Model-agnostic Explanations (LIME) is a typical method in explainable machine learning. In 2016, Ribeiro, Marco Tulio *et al.* proposed LIME, which referred to an algorithm designed to explain machine learning models, by replacing a non-explainable model with a simple explainable local model, providing explainability and confidence assessment of model predictions [166]. Any black-box model can use this algorithm. In LIME, a input data $z = (z_1, z_2, ..., z_{N_{exp}})$ point is perturbed with domain to get $zp = (zp_1, zp_2, ..., zp_{N_{exp}})$, where N_{exp} is the number of features. A set zp_{i}^* can be obtained through repeated iterations. The output $f_{nim}(zp_1^*, zp_2^*, ..., zp_{N_{exp}}^*)$ can be obtained through the neural network f_{nim}, that is the network to be explained. Then, a simple explainable neural network f_{im}, such as linear regression or ridge regression, can fit these perturbed input-output pairs. The loss function is shown in

$$L(f_{nim}, f_{im})_{\mathrm{lime}} = \sum_{i=1}^{N_{exp}} w_i \cdot (f_{nim}(zp_i^*) - f_{im}(zp_i^*)), \tag{2.20}$$

$$w_i = \mathrm{e}^{(-\zeta \cdot distance(z, zp))}, \tag{2.21}$$

where the formula (2.20) aims to replace the non-explainable model f_{nim} with an explainable model f_{im} in a weighted manner, where the weights are denoted by w_i as expressed in formula (2.21) and w_i is in [0, 1]. Here, ζ is a hyperparameter, and a larger value of ζ will rapidly decrease the weights of samples farther away. The specific range can be determined based on the problem's characteristics and

the dataset's distribution. The function *distance* used for distance weighting can be any distance function, such as L2 distance or cosine similarity.

The weights of the simple neural network f_{im} are then used for explanation and locating the key elements of the input features with domains. The larger the weight, the more important the feature is, indicating its significance to the output.

In addition to LIME, in 2015, Bach, Sebastian *et al.* proposed the LRP (Layer-Wise Relevance Propagation) algorithm, which was a method used to explain the decisions made by nonlinear classifiers [167]. It visualized the contributions of individual pixels to classification predictions through pixel-level decomposition, typically applied in the context of image classification and model understanding.

In 2017, Selvaraju, Ramprasaath R *et al.* proposed Grad-CAM (Gradient-weighted Class Activation Mapping) as a technique for generating visual explanations of CNN models, highlighting important regions in images for understanding models [168].

In 2017, Lundberg, Scott M *et al.* proposed the SHAP (Shapley Additive exPlanations) algorithm, a method to explain the predictions of machine learning models by assigning importance values to each feature and reveal their contributions to the results [169].

Additionally, there were many applications of the explainable machine learning algorithms, such as in the finance and healthcare sectors [170] [171] [172] [173] [174].

Interpretable machine learning

From the perspective of explainability, 'Interpretable' is more potent than 'Explainable', as 'Interpretable' refers to white-box models that can be explained through each step of reasoning in tree or graph structures. In contrast, 'Explainable' is used to interpret black-box models, and its explainability is not as stable as that of white-box models' interpretable.

In 2017, Das, Rajarshi *et al.* proposed Meandering In Networks of Entities to Reach Verisimilar Answers, which utilized a reinforcement learning approach to address knowledge graph completion problems, yielding inference paths. The paths themselves offered some degree of interpretation [175].

In 2017, Xiong, Wenhan *et al.* proposed a novel reinforcement learning framework, DeepPath, for learning multi-hop relational paths and its reasons within the vector space of the knowledge graph [176]. Similar to [175], the reasoning paths served as the interpretation.

In 2018, Ai *et al.* proposed a knowledge graph-based interpretation framework [177]. Based on the embedded knowledge base, a soft matching algorithm was developed, using the inner product of embeddings for personalized interpretations of recommended items. On e-commerce datasets, the simulations revealed superior recommendation performance and interpretation power.

In 2019, Wang, Xiang *et al.* proposed a new model named Knowledge-aware Path Recurrent Network [178]. This algorithm incorporated a weighted pooling operation to distinguish the strengths of different paths in connecting users and items, effectively inferred the underlying rationale of user-item interactions, thereby endowing the model with interpretation.

In 2022, Geng, Shijie *et al.* proposed a framework named Path Language Modeling Recommendation [179]. This algorithm learned a language model over the paths in the knowledge graph, including entities and edges, and achieved recommendation and interpretation simultaneously through path sequence decoding.

As mentioned above, interpretations are typically provided by selecting paths or assigning weights to each path in the knowledge graph. All the related algorithms in explainable and interpretable machine learning are summarized in table 2.14.

2.5 Distance functions and metrics

This section introduces some mathematical concepts, including L2 distance, cosine similarity, Kendall coefficient, and Pareto front. These concepts will be used in the following Chapters.

L2 distance and Cosine similarity

L2 distance, also known as the Euclidean distance, measures the straight-line distance between two points in Euclidean space. The L2 distance between two points

Table 2.14: Literature review on explainable and interpretable machine learning algorithms.

Year	Reference	Field	Name	Based on	Contribution
2015	Bach, Sebastian *et al.* [167]	Explanable machine learning	Layer-wise relevance propagation	Backpropagation	Visualized the contributions of individual pixels to classify predictions through pixel-level decompositions.
2016	Ribeiro, Marco Tulio *et al.* [166]	Explainable machine learning	Local interpretable model-agnostic explanations	Explainable models such as linear regression	Used to explain the predictions of models by approximating them locally with interpretable models.
2017	Selvaraju, Ramprasaath R *et al.* [168]	Explainable machine learning	Gradient-weighted class activation mapping	Convolutional neural network	Designed for visualizing the important regions of input for convolutional neural networks.
2017	Lundberg, Scott M *et al.* [169]	Explanable machine learning	SHapley additive exPlanations	Cooperative game theory of Shapley values	Explained the output of models using game theory, quantifying each feature's contribution to the prediction.
2017	Das, Rajarshi *et al.* [175]	Interpretable machine learning	Meandering in networks of entities to reach verisimilar answers	Reinforcement learning	Utilized a reinforcement learning approach, yielding inference paths on knowledge graphs.
2017	Xiong, Wenhan *et al.* [176]	Interpretable machine learning	DeepPath	Reinforcement learning	Generated multi-hop relational paths in knowledge graphs with reinforcement learning.
2018	Ai, Qingyao *et al.* [177]	Interpretable machine learning	Embed heterogeneous entities for recommendation	Graph embedding	Utilized knowledge-base embeddings and a soft matching algorithm for interpretable recommendations.
2019	Wang, Xiang *et al.* [178]	Interpretable machine learning	Knowledge-aware path recurrent network	Long-short term memory	Inferred the underlying rationale of user-item interactions by leveraging the sequential dependencies within a path.
2022	Geng, Shijie *et al.* [179]	Interpretable machine learning	Path language modelling recommendation	Transformer	Integrated personalized recommendation systems with the Transformer algorithm to provide credible suggestions and corresponding interpretions.

p_1 and p_2 is defined by the formula

$$\mathrm{L2}(p_1, p_2) = \sqrt{\sum_{i=1}^{I} (p_{1_i} - p_{2_i})^2}. \tag{2.22}$$

Cosine similarity is a commonly used method for measuring the similarity between two non-zero vectors based on their direction. Unlike Euclidean distance, cosine similarity focuses on the difference in angle between vectors rather than their magnitude, making it suitable for similarity calculations in text data and multidimensional spaces. The formula for calculating cosine similarity is

$$\cos(\theta) = \frac{p_1 \cdot p_2}{\|p_1\| \|p_2\|} = \frac{\sum_{i=1}^{I} p_{1_i} p_{2_i}}{\sqrt{\sum_{i=1}^{I} p_{1_i}^2} \sqrt{\sum_{i=1}^{I} p_{2_i}^2}}. \tag{2.23}$$

Kendall coefficient

The Kendall coefficient is a statistical method to measure the similarity of the orderings between two variables, especially to quantify the degree of consistency in the rankings of two variables. The Kendall coefficient is shown as

$$\mathrm{Kendall} = \frac{\beta - \psi}{\mathrm{length}(\mathrm{length} - 1)/2}, \tag{2.24}$$

where β and ψ are the orderly and disorderly rankings of the data sequences, the length is the number of statistical objects.

Pareto front

The Pareto front is the set of all non-dominated solutions in bi-objective optimization [180]. Consider a system with the function $\mathrm{f} : \mathbb{PA} \to \mathbb{R}^M$, where $\mathbb{PA}$ is a set of feasible decisions in the metric space $\mathbb{R}^M$ and $\mathbb{P}$ is the feasible set of criterion vectors in $\mathbb{R}^M$, such that $\mathbb{P} = p = f(pa), \forall\ \mathrm{pa} \in \mathbb{PA}$. If a point p_1 strictly dominates another point p_2, written as $p_1 \succ p_2$. The Pareto frontier $F_{\mathrm{pareto}}(\mathbb{P})$ is thus written as

$$F_{\mathrm{pareto}}(\mathbb{P}) = \{p_2 \in \mathbb{P} : \{p_1 \in \mathbb{P} : p_1 \succ p_2, p_1 \neq p_2\} = \emptyset\}\ . \tag{2.25}$$

2.6 Research gaps

Based on the introduction and literature review, the research gaps can be summarized as follows.

2.6.1 Household device action recommendation system

In the existing literature, some recommendation systems have a low accuracy when making recommendations, failing to make precise recommendations and not being able to capture and predict a user's behavior accurately [17] [18]. The primary issue is to model a users' daily appliance usage and provide accurate recommendations for the next household device action. After this, the recommendation results will closely relate to people's lives, thus the model's interpretability is essential for supporting the AI recommendation application in homes, and provides researchers feedback to improve the recommendations' performance. However, as shown in table 2.1 and 2.2, most smart home systems do not consider interpretability. Finally, improving energy efficiency is also required, however, few studies link electrical optimization issues with household device action recommendations. Building the connections between recommendation lists and demand response algorithms is an interesting direction for research (The recommendation system's results are often a probabilistic list, reflecting the possibility of using each appliance in the next action. Using the list of expected appliance actions derived from the recommendations, a bi-objective optimization can be performed on both the expected power consumption and the expected satisfaction of users from the recommendation list.).

Nevertheless, according to the summary in table 2.1 and 2.2, no existing paper has considered all of these aspects (Including energy optimization, personalization, contextual awareness, user satisfaction model, location and interpretability). These research gaps will be filled by the proposed framework in Chapter 3.

2.6.2 Home microgrid scheduling systems

In the existing literature, most scheduling algorithms only focus on optimizing cumulative electricity costs, and not utilizing real-time scheduling methods (DQN is

capable of real-time learning through interaction with the environment, continuously adjusting and optimizing its strategy based on feedback, and adapting to new environments or changes. However, optimization algorithms such as genetic algorithms typically require the completion of one iteration before evaluating and generating the next generation, and when the environment changes, they may need to be re-optimized or have their parameters adjusted. In this thesis, 'real-time' refers to the ability to make decisions in near real-time when the environment changes or in a dynamic environment, typically within 5 minutes. Optimization algorithms such as genetic algorithms often cannot achieve that.) [4]. Establishing a real-time bi-objective optimization algorithm poses a challenge; For example, optimizing both accumulated carbon emissions and electricity costs in real-time is a challenge, furthermore, the optimization goals could contradict each other. Additionally, as user load data is utilized, there is a need to address the protection of personal privacy whilst leveraging the data from different households. However, as shown between table 2.3 and 2.5, most papers on power system optimization do not consider data privacy protection. Lastly, achieving model adjustability with orientation when performing bi-objective optimization is a research goal. For example, choosing the appropriate bias between carbon emissions and electricity costs optimizations.

Nonetheless, according to the summary in tables 2.3, 2.4, and 2.5, these papers jointly have failed to consider energy optimization, carbon optimization, privacy protection as well as model adjustability. These research gaps will be filled by the proposed algorithm in Chapter 4.

2.6.3 Privacy protection

Federated learning is widely used in household distributed machine learning scenarios, effectively utilizing data from various parties (clients) while protecting privacy [75]. Nevertheless, research indicates that the gradients in federated learning contain private information from the training data, which could potentially be maliciously attacked and obtained without the user being aware of the security issue [114] [119] [115] [116] [117] [118]. Hence, a topic that requires further research is predict to what extent federated learning protects a residents' privacy. The key fac-

tors that could influence privacy leakage in federated learning should be identified. Additionally, the possible methods to prevent these attacks and any subsequent privacy leakage in federated learning should be researched. However, these potential risks have not been considered in federated electrical scenarios, such as in smart homes and home microgrids. The proposed framework in Chapter 5 will fill these research gaps.

2.7 Conclusion

Section 2.1 (the Smart Communities section) offers a literature review on the content covered in Chapters 3 to 4, including smart homes and home microgrids and their applications (household device action recommendation systems as well as DRL-based microgrid scheduling algorithms). Following this, section 2.2 reviews federated learning, its applications in the power grid, and any potential attacks on federated learning, laying the groundwork for the algorithms to be used in Chapter 5. Finally, section 2.3 overviews deep learning algorithms, introducing literature specific to each type of algorithm. These algorithms will be used from Chapter 3 to Chapter 5.

CHAPTER 3

Domestic device action recommendation system

With the research gaps summarized in Chapter 2, this chapter focuses on a household device action recommendation system. The first goal of this chapter is to establish a smart home device action recommendation system, enabling artificial intelligence to understand human habits and behaviors better. The next objective is to interpret the recommended results of the device action, making it understandable for the users. Lastly, to develop a demand response algorithm based on action recommendation results. In this chapter, 'demand response' has the meaning that residents can balance personal expected satisfaction against power consumption by setting the appropriate power level for each home appliance, thus improving overall energy efficiency. 'Personal expected satisfaction' refers to the anticipated level of dissatisfaction a user expects to experience due to the delay in using an appliance [181].

A recommendation framework called DARK (Device Action Recommendation System with Knowledge Graph) is proposed. The overall architecture is designed and built upon a household appliance system, as is shown in Fig. 3.1. The system comprises of appliances, sensors, a small-scale database, and the DARK algorithm on the right side. All the appliances are connected to the local area network of the

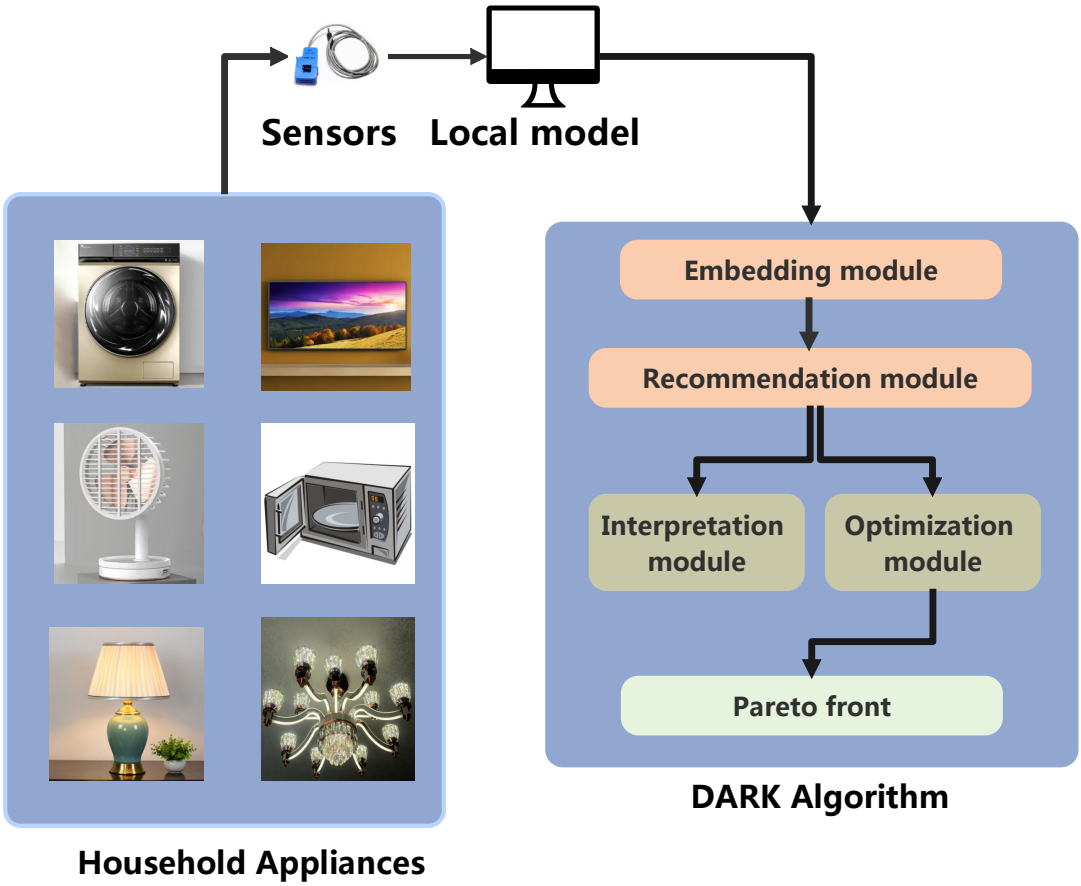

Figure 3.1: The proposed system architecture of DARK has different modules. The energy consumption information is collected by sensors and used in the local model. Then, the data is sent to different modules, including the embedding, recommendation, interpretation and optimization modules.

home, and these are monitored by the sensors to collect the energy consumption data. The advantage of this data is that it not only represents the consumption information but also it indicates the existence of the operational data for the appliance, such as the status of switches and modes. Please note that this chapter only considers electricity consumption, and no other kinds of consumption such as geothermal or gas. The knowledge graph is formed by processing the data in the local database, and then the knowledge graph is encoded to embedding in the embedding module. The embedded knowledge graph is used for making recommendations in the recommendation module. Next, the recommended results are sent to the interpretation module to generate interpretations and to the optimization module to generate optimization results for the user's expected satisfaction and energy consumption, assisting users in decision-making.

The DARK algorithm selected KGAT as the recommendation algorithm and made improvements to it. The reasons for choosing this algorithm are as follows: 1. KGAT integrates well with graph data and often performs better than various common deep learning algorithms in simulations when dealing with graph data. 2. The KGAT algorithm offers model interpretability, allowing for explanations to be generated through its graph paths, and the attention mechanism can help assess the importance of these paths.

Privacy protection is also essential. User data is considered private and should not be disclosed to third parties. To ensure user privacy, the database data is not shared externally, instead federated learning can be employed, only uploading the local model to the server. Federated learning ensures that the data remains only locally stored [75]; however, this is not the focus of this chapter. In this chapter, only standalone action recommendation systems are considered, without sharing any data, so there is no privacy leakage concerns in this chapter. For further details on privacy protection, please refer to Chapter 5.

3.1 Graph construction

This section introduces the preparatory work of DARK framework, including the data collection and the steps to construct a knowledge graph.

3.1.1 Domestic device data collection

In human daily activities, a significant amount of electrical usage data, appliance data, electricity consumption time data, appliance location data, and common knowledge are generated [11]. These data intertwine to form a network and can be modelled as a graph. Graph models can represent the relationships between different appliances. This contributes to a better understanding of the interactions among household appliances. The following data on appliances are collected for the DARK framework:

1) Rated Power: Each appliance has its rated power; for example, a toaster has a rated power of 1500 Watts, while the power of a light is generally between 20 and

50 Watts.

2) Appliance Location: Appliances are distributed across different rooms, including the kitchen, living room, office, bedroom, utility room, children's room and undefined locations. If the location is not defined, it is assumed the appliance is in a unique location.

3) The impact on user satisfaction: Each appliance has a different impact on the expected user satisfaction. For instance, if delayed to use, washing machines have a lower impact due to the demand response ability, whereas appliances like microwaves or gas ovens have a higher impact.

4) Habitual Usage Time: Each appliance has specific habitual usage times. For example, gas ovens are most used during mealtime, while lights are frequently used at night.

5) Habitual Usage Sequence: Some appliances have a habitual usage sequence. For instance, the most commonly used appliance following a dryer might be a hair straightener.

All data is organized into a knowledge graph, in 'head entity' - 'relationship' style - 'tail entity', such as 'dishwasher' - 'operates from' - '18:00-20:00'. These knowledge graph data together form a graph, which will be sent to the Embedding module.

3.1.2 Building knowledge graph

It is assumed that each appliance has an associated sub-meter, so individual electricity usage data for each appliance can be obtained. If this is not achievable and only a main smart meter exists, non-intrusive load monitoring can be used for power disaggregation to obtain individual appliance electricity data. Then the next step would be data processing for DARK, involving the following sequence of steps: format conversion, noise filtering, device selection, graph construction from conventional datasets, incorporating other attributes, and finally negative sampling.

Format conversion and noise filtering

Including converting the data into a format that Python can recognize, transforming the timestamps into year-month-day format, and aggregating the data into certain

time intervals. The data may contain a large amount of noise, so filters are implemented to determine the on/off states of appliances and filter out momentarily switched-on and electricity leakage. Due to the unique characteristics of each type of electrical appliance, the filtering thresholds for each type of appliance need to be customized according to the rated power.

Domestic devices selection

Some appliances are not suitable for inclusion in the recommendation system. As shown in the Fig. 3.2 and Fig. 3.3, these graphs display the typical power consumption of the fridge and washing machine for a day (24 hours) [182]. The fridge exhibits characteristics of a regular cycle of operation, making it less suitable for recommendation system data. Please note that even after filtering, the refrigerator still exhibits significant periodic high peaks. These are not detailed in the UK-DALE dataset, but they can be assumed from the opening and closing of the refrigerator or the refrigerator's periodic defrost function. On the other hand, the washing machine shows distinct triggering patterns, making it a suitable candidate for inclusion in the recommendation system data. So appliances with long-term operation, such as refrigerators, routers, iPad chargers, etc., are excluded. Additionally, some appliances that are not under human control also need to be excluded. For example, the function of boilers is based on a preset automatic control system.

Graph construction

Each appliance is treated as a node, and an edge between two appliances is added if they have a sequential relationship, labelled as "used before/after." The time zone knowledge of appliances is added, connecting time nodes with appliance nodes. The relationship can be represented as "used during the time period". In addition, the location information of appliances is established, connecting location nodes with appliance nodes and labelling the relationship as "located in the room". Other attributes, such as the rated power and the user's expected satisfaction coefficient, are also added for each appliance.

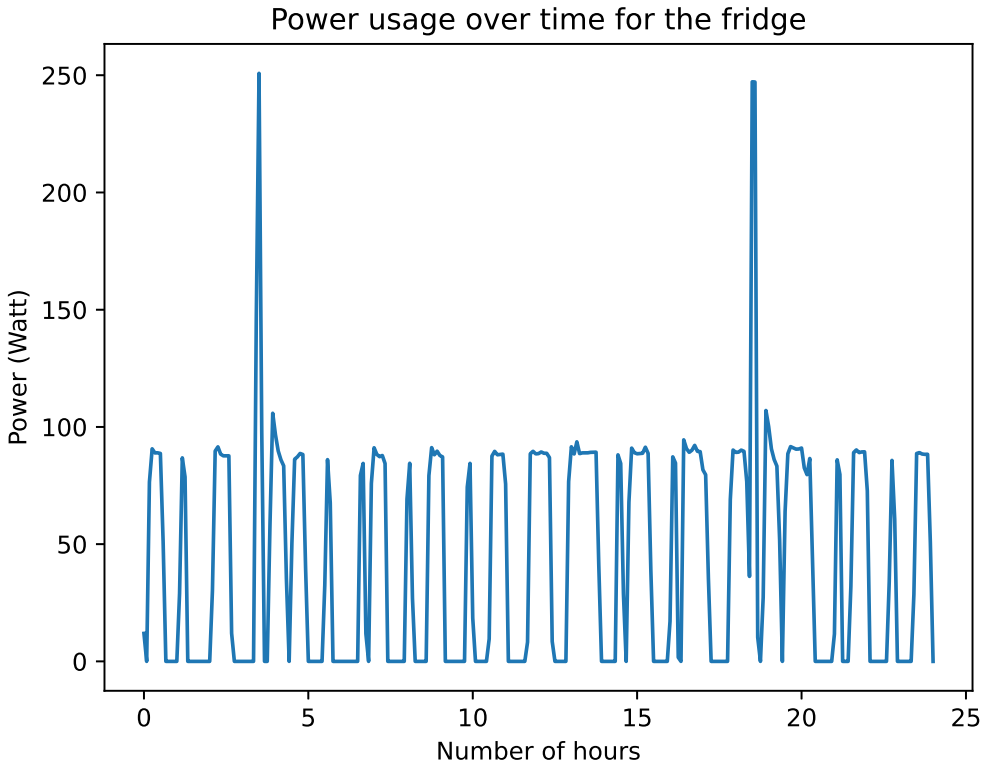

Figure 3.2: Power consumption for the fridge over time. It can be seen in the graph that it is used almost every hour, and because of its intensive use, this type of appliance is not suitable for recommendation systems.

Negative sampling

Negative sampling is performed on the dataset to accelerate convergence. For example, a gas oven is often used from 18:00 to 24:00, recorded as 'gas oven-used during the time-18:00'. This time can be replaced by 2:00 to 4:00 in the morning for producing negative examples, which has never happened, recorded as 'gas oven-never used during the time-3:00'. This method creates negative examples at a 1:1 ratio in numbers with positive examples.

Compared to conventional data, by modelling the relationships between nodes in graph data and visualizing the connections between household data nodes and edges, one can intuitively understand how the proposed KGAT model performs reasoning and makes recommendations. This also imparts a structural nature to the model's decisions, beyond merely relying on the statistical properties of the data, which facilitates data mining [183].

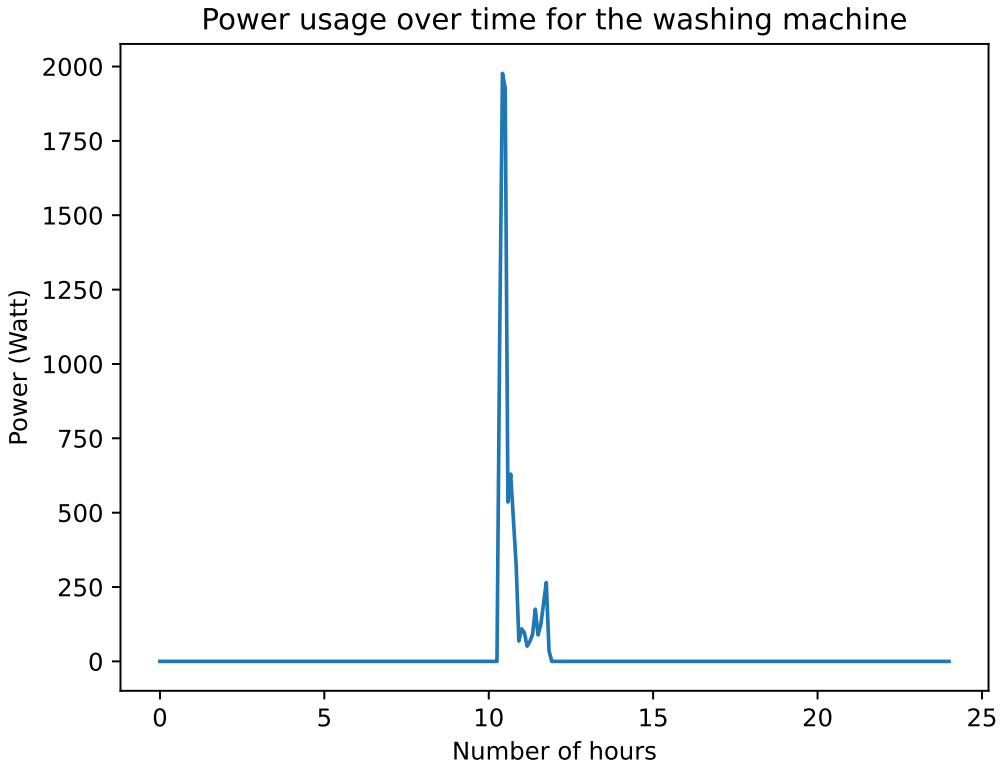

Figure 3.3: Power consumption for the washing machine over time. The graph shows distinct electricity usage characteristics between 10 and 12 hours, making this appliance suitable for recommendation systems.

3.2 Embedding module

During the data processing in the embedding module of DARK, it is necessary to convert the data into embeddings, which use high-dimensional vectors to represent the discrete data. The transformation from data to embeddings facilitates the aggregations of knowledge graph algorithms in subsequent steps.

The embedding layer aims to train the embedding of (h, r, t) such that the embedding of the head em_h plus the embedding of relation em_r approach the embedding of tail em_t for any positive (h, r, t) in distance (for example L2 distance), which can be represented as [184]

$$g_{(h,r,t)} = || W_r em_h + em_r - W_r em_t||_2^2 \ , \tag{3.1}$$

where $|| \cdot ||_2^2$ means the squared results of the L2 distance. W_r is the relation weight

matrix, it was defined in TransR [184], helping to map the heads and tails from the original entity space to the relation space, however it can also be ignored in some embedding algorithms like TransE. $g_{(h,r,t)}$ represents the positive triples and the value of $g_{(h,r,t)}$ should be close to zero. However, for negative triples, which means the h, r or t can be replaced by other values that never happened. For example, $g_{(h,r,t_{\mathbb{D}})}$ is generated by replacing t with $t_{\mathbb{D}}$ arbitrarily and the value of $g_{(h,r,t_{\mathbb{D}})}$ should be infinity. In the proposed recommendation system, negative sampling was applied to each of the data types mentioned above, and the training was conducted accordingly.

During backpropagation, its parameters are updated based on the gradient of the loss function to minimize the difference between predicted results and true labels. The following data was encoded to embeddings:

1. Appliances: Each appliance is assigned an embedding, resulting in a set of embeddings for different appliances.

2. Time zones: The daily time is divided into four time periods: midnight, morning, afternoon, and evening. Thus, a week can be divided into 4×7=28 time zones. These 28 time zones are encoded as a set of embeddings, providing information about a specific day and time within a week.

3. Locations: Appliances are distributed across different rooms, and an embedding is created for each room.

The embedding module processes the input data, which converts discrete inputs into continuous embeddings. These embeddings are then fed into the knowledge graph layer as input for training.

3.3 Recommendation module

This section introduces the proposed DARK algorithm, a improved KGAT algorithm in the recommendation area, and its time complexity are analysed. It also includes a comparison with the traditional KGAT algorithm.

3.3.1 Proposed DARK algorithm

This section mainly discusses the customized KGAT algorithm for the home appliance recommendation system in the recommendation module of DARK, with improvements made to the original KGAT algorithm.

Algorithm 1: The proposed DARK algorithm with improved next-action recommendation scenario

Load sequence embeddings $em^{se}_{(h,r,t,\text{time})}$ from processed datasets.
Load time zone embeddings $em^{ti}_{(h,r,t)}$ from processed datasets.
Load location embeddings $em^{lo}_{(h,r,t)}$ from processed datasets.
Initialize the data distribution from the datasets.
Initialize the weight of KGAT including W_r and W_u.
Initialize a fully connected layer FullyConnectedLayer.
Initialize the list $\mathbb{L}$.
for *episode* $= 1, M$ **do**
    for em^{se}_h, em^{se}_t *in* $em^{se}_{(h,r,t,time)}$ **do**
        Sampling dataset s_h for em_h and s_t for em_t according to the distribution of $em^{se}_{(h,r,t)}$, $em^{ti}_{(h,r,t)}$ and $em^{lo}_{(h,r,t)}$.
        Constructing adjacency matrices $\mathbf{Adj}_h$ with s_h and $\mathbf{Adj}_t$ with s_t.
        for *each* em_h, em_r *and* em_t *in* $\mathbf{Adj}_h$ **do**
            $atn_{h\text{-}(h,r,t)} =$
            $\text{Softmax}((W_r em_t)^{\mathrm{T}} \tanh(W_r em_h + em_r))$
            $em_{h\text{-}Nh} = \sum_{(h,r,t)\in Nh} atn_{h\text{-}(h,r,t)} \times em_t$
            $agg_h = \text{LeakyRelu}(W_u(em_h + em_{h\text{-}Nh}))$
        end for
        for *each* em_h, em_r *and* em_t *in* $\mathbf{Adj}_t$ **do**
            $atn_{t\text{-}(h,r,t)} =$
            $\text{Softmax}((W_r em_t)^{\mathrm{T}} \tanh(W_r em_h + em_r))$
            $em_{t\text{-}Nh} = \sum_{(h,r,t)\in Nh} atn_{t\text{-}(h,r,t)} \times em_t$
            $agg_t = \text{LeakyRelu}(W_u(em_h + em_{t\text{-}Nh}))$
        end for
        $Vec = \text{concatenate}(agg_h, agg_t, em^{se}_{\text{time}})$
        Store $O = \text{Relu}(\text{FullyConnectedLayer}(Vec))$ in $\mathbb{L}$
    end for
end for
Output: $\mathbb{L}$

Different from the traditional KGAT algorithm, the following DARK algorithm improvements have been customized for the next-action recommendation scenario. The algorithm's pseudocode is shown as **Algorithm 1**, and the related algorithm flow is shown in figure 3.4.

Firstly, the knowledge graph algorithm establishes an adjacency matrix based on

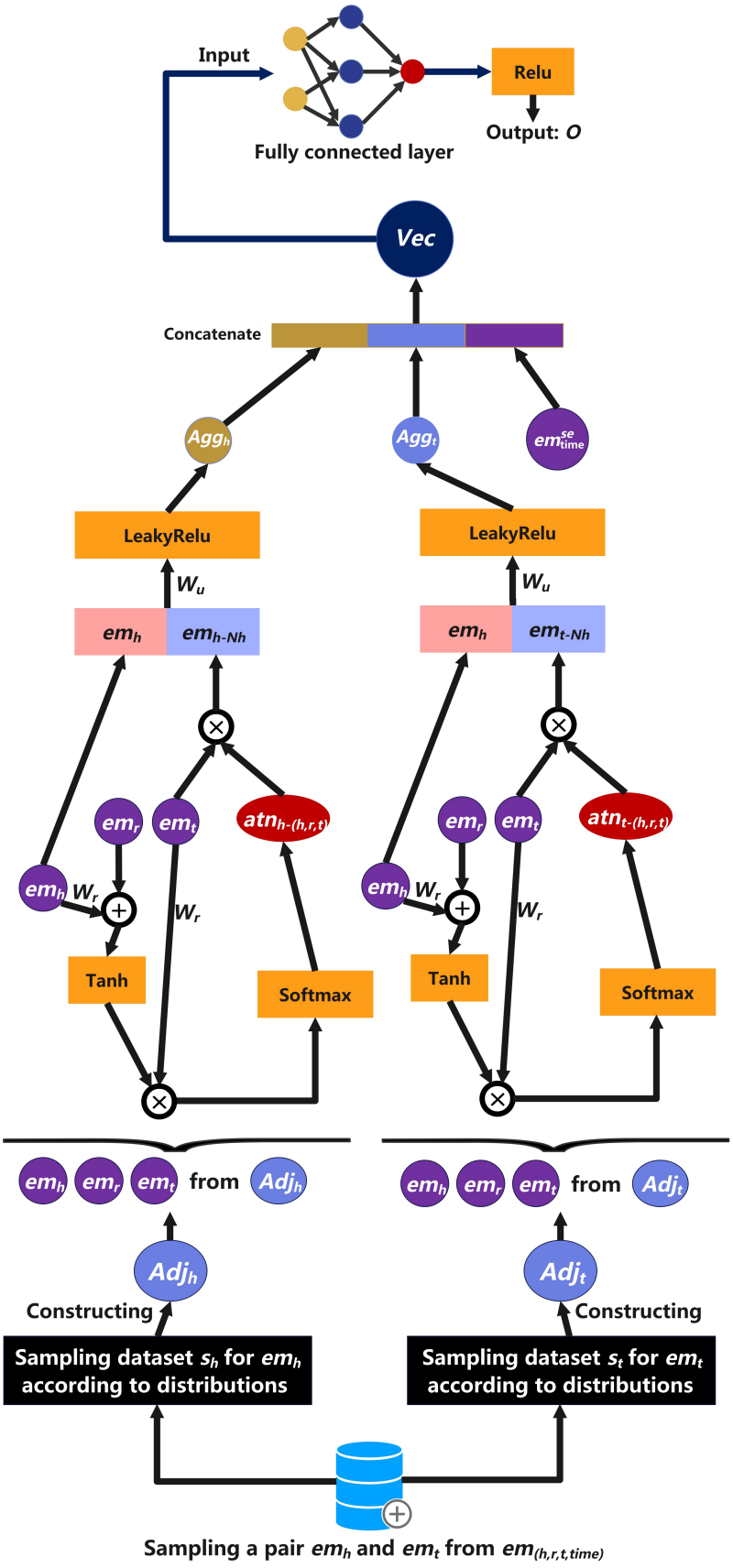

Figure 3.4: The diagram of the proposed DARK algorithm.

randomly sampled neighbouring nodes for embedding aggregation. Each attribute type, such as timezone, usage sequence, and location, is collected in the modified algorithm, ensuring equal chances of different kinds of data. After that, random sampling is replaced with probability-based sampling according to the distribution. These samples will inform the knowledge graph algorithms about the studied case and affect the aggregation results. For example, if a particular appliance is most commonly used on Tuesday mornings, the probability of sampling Tuesday mornings for the adjacency matrix will be higher. This is expressed in lines 5-6 of the **Algorithm 1**.

In addition, the algorithm aims to predict the next potential appliance based on residents' appliance usage habits. In the traditional KGAT recommendation algorithm, only candidate (potentially recommended) appliances undergo graph embedding aggregation because conventional KGAT treats the recommended appliance as an item and the known appliance as a user. However, in the proposed problem, the known appliance usage can also be treated as an item and aggregated. This is expressed in the line 13-18 of the **Algorithm 1**. Lines 13-14 calculate the attention term $a_{h\text{-}(h,r,t)}$ in KGAT, line 15 obtains the first aggregated embedding vector $em_{t\text{-}Nh}$ based on the attention mechanism, and line 16 uses a deep neural network to compute the final aggregated vector agg_h.

The output $\mathbb{L}$ contains aggregated embedding agg_h for head embeddings and agg_t for tail embeddings. Then the embedding $\mathbb{L}$ is used as the input of the fully connected layers for training in line 20 of the **Algorithm 1**.

Fully connected layers are used as the subsequent classifier, outputting a value between 0 and 1 to determine whether two appliances will be used continuously. The closer the value is to 1, the higher the probability of continuous usage.

3.3.2 Algorithm time complexity

Algorithm time complexity is a measure of algorithm efficiency, indicating the relationship between the running time of an algorithm and the size of its input. It is crucial for evaluating algorithm performance and making improvements.

1) Embeddings: For knowledge graph embedding, according to the translation

principle of (3.1), the algorithm time complexity is $\mathbb{O}(G \times d^2)$, where G represents the size of the knowledge graph, and d is the embedding dimension [150].

2) KGAT layer: In the attention embedding propagation of KGAT, the algorithm time complexity of the matrix multiplication for the a-th layer is $\mathbb{O}(G \times d_a \times d_{a-1})$, where d_a and d_{a-1} are the current and previous layer dimensions [150].

3) Proposed KGAT layer: In addition to the KGAT layer, due to the need for sampling for each category, an algorithm time complexity $\mathbb{O}(1)$ is added, which can also be ignored. In the attention embedding propagation, another $\mathbb{O}(G \times d_a \times d_{a-1})$ is added. So the total algorithm time complexity is $\mathbb{O}(2 \times G \times d_a \times d_{a-1})$ and the 2 can also be ignored according to the expression of the algorithm time complexity [150].

4) DNN layer: After the KGAT layer, one or more DNN layers need to be connected as classifiers. For DNN, the algorithm time complexity is $\mathbb{O}(d_a \times d_{a-1})$, similarly, d_a and d_{a-1} the dimension of the input/output channel of DNN [185].

In summary, the proposed algorithm does not introduce additional terms that increase the dimensionality of algorithm time complexity.

3.4 Interpretation module

In the interpretation module of DARK, when generating action recommendations, interpretability is crucial for making the recommendation results convincing. The purpose of this module is to provide further interpretations for the recommended results. It has been known that the trained embeddings in graph neural networks can represent the meaning of the data [147], and in paper [177], Ai *et al.* used that for interpretation. Based on this characteristic, reasons for the recommended results are generated, which consist of three types of reasons:

1. Habitual usage sequence means an appliance is often used after another.

2. Habitual usage time period means that two appliances are often used at the same specific time period.

3. Usage at a habitual location means an appliance is used because it has location relationships with other appliances.

All the connections between the recommended appliances and the known appliances within two hops are identified in the graph, and the embeddings of all the nodes and relationships involved are found. Using TransE and the approach described in the paper [147] in (3.2), the head embedding plus relation embedding should approach the tail embedding as much as possible after training. Ideally, they are equal.

$$em_t \approx em_h + em_r \ . \tag{3.2}$$

Here, a new measure is shown in (3.3)-(3.8) for the recommendation results in the context of household appliance action recommendations. From (3.2), the embedding of the reasons can be calculated in (3.3)-(3.5).

$$em^{se}_{\text{total}} = (em^{se}_h + em^{se}_r)em^{se}_t \ , \tag{3.3}$$

$$em^{ti}_{\text{total}} = (\sum_{i=1}^{N}(em^{ti}_{h_i} + em^{ti}_{r_i})em^{ti}_{t_i})/N \ , \tag{3.4}$$

$$em^{lo}_{\text{total}} = (em^{lo}_h + em^{lo}_r)em^{lo}_t \ , \tag{3.5}$$

$$R_{se} = \frac{em^{se}_{\text{total}}}{em^{se}_{\text{total}} + em^{ti}_{\text{total}} + em^{lo}_{\text{total}}} \ , \tag{3.6}$$

$$R_{ti} = \frac{em^{ti}_{\text{total}}}{em^{se}_{\text{total}} + em^{ti}_{\text{total}} + em^{lo}_{\text{total}}} \ , \tag{3.7}$$

$$R_{lo} = \frac{em^{lo}_{\text{total}}}{em^{se}_{\text{total}} + em^{ti}_{\text{total}} + em^{lo}_{\text{total}}} \ , \tag{3.8}$$

where the superscripts se, ti and lo refer to different reasons like sequence, timezone, and location. The value em^{se}_{total}, em^{ti}_{total} and em^{lo}_{total} are the inner product value of the embeddings. From the characteristic embedding of TransE [177], a higher product value means the reason has a higher relationship with the answer, where em^{ti}_{total} is the top N averages product of embedding of the reason timezone. em^{se}_{total} and em^{lo}_{total} only exist if two household appliances have a sequential relationship or are placed in the same location. In (3.6)-(3.8), the explain R of different reasons is calculated.

With the above method, the recommendation was interpreted based on the inner product of embeddings. Each recommendation can be interpreted as combining three reasons with different percentages.

Please note that a major drawback of interpretation is that, generally, the correctness of the interpretation cannot be verified, thus it is usually used as a reference rather than a precise interpretation. In this thesis, the verification method only checks whether the interpretations align with common sense. If this is true, the interpretations are considered to be correct. Since interpretations themselves are estimations of the user's intentions, surveys can be created in the future to further verify the accuracy of these interpretations.

3.5 Optimization module

This section primarily introduces the objective functions in the optimization module of DARK, for optimizing energy consumption and user's expected satisfaction.

It is assumed that the power of each appliance is adjustable within a certain range, and the power of an appliance affects the user's expected satisfaction as given by

$$f_{\text{satisfaction}} = -f_{\text{dissatisfaction}} = -\sum_{i=1}^{N} df_i (P_i - P_i^{\text{real}})^2 \ , \tag{3.9}$$

where $f_{\text{satisfaction}}$ is a user's expected satisfaction function, and $f_{\text{dissatisfaction}}$ represents the sum of dissatisfaction scores for N appliances. The rated power of appliance i is P_i^{real} with $i = 1, \ldots, N$. Power significantly above or below the rated power can result in dissatisfaction of users. Factor df_i is the dissatisfaction factor. The averaged power load of appliances is calculated by

$$f_{\text{avgpower}} = \sum_{i=1}^{N} P_i^{\text{real}} \ . \tag{3.10}$$

For (3.9) and (3.10), two competing objectives are considered: maximizing user's expected satisfaction, which represents consumer needs, and minimizing average power load, a consideration for saving power grid consumptions. For this type of optimization problem, multi-objective optimization algorithms, such as genetic algorithms, can be used to find the Pareto front. The outcome of the optimization is to set power level P_i for appliance i based on the expectation of the recommendation results. Through the optimization results, multiple feasible solutions between user's

expected satisfaction and average power load can be selected in the Pareto front [186].

3.6 Simulation setup

This section introduces the simulation environment, dataset preprocessing and the recommendation results, including accuracy and interpretations.

The PyTorch framework was used for software to build the neural network architectures [9], including DNN, CNN [131], and RNN [132]. KGAT [150] and proposed KGAT were built based on the DNN toolbox and matrix calculations. The genetic algorithm toolbox from Geatpy was utilized to handle optimization problems [187].

For hardware, 16GB NVIDIA GeForce RTX 3080 GPU was used with Intel i7-12700H CPU and 32GB memory.

3.6.1 Dataset and data preprocessing

For datasets, UK-DALE (UK Domestic Appliance-Level Electricity) is a publicly available household electricity dataset used for power consumption monitoring and non-intrusive load recognition research [182]. The UK-DALE dataset provides high-precision power consumption data, including 54 household appliances such as refrigerators, washing machines, ovens, lighting, etc. It is based on electricity-triggered information. The detailed sampling frequency, reaching as high as 6 Hz, makes constructing the graph's structure suitable. Manual construction of the knowledge graph is performed in DARK based on UK-DALE.

54 types of appliance power consumption data have been collected in the UK-DALE dataset. After manual filtering, 37 out of the 54 selected appliances were chosen for simulation.

Fig. 3.5 illustrates the constructed knowledge graph, which is a part of the overall knowledge graph. It depicts a subgraph containing a kitchen light, microwave, and kettle, each appliance's attributes. For instance, the most commonly used time for the microwave is 6:00 to 12:00 on Tuesday. Similarly, the rated power of the Kettle is 2000 Watts. There are also associations between appliances. For example,

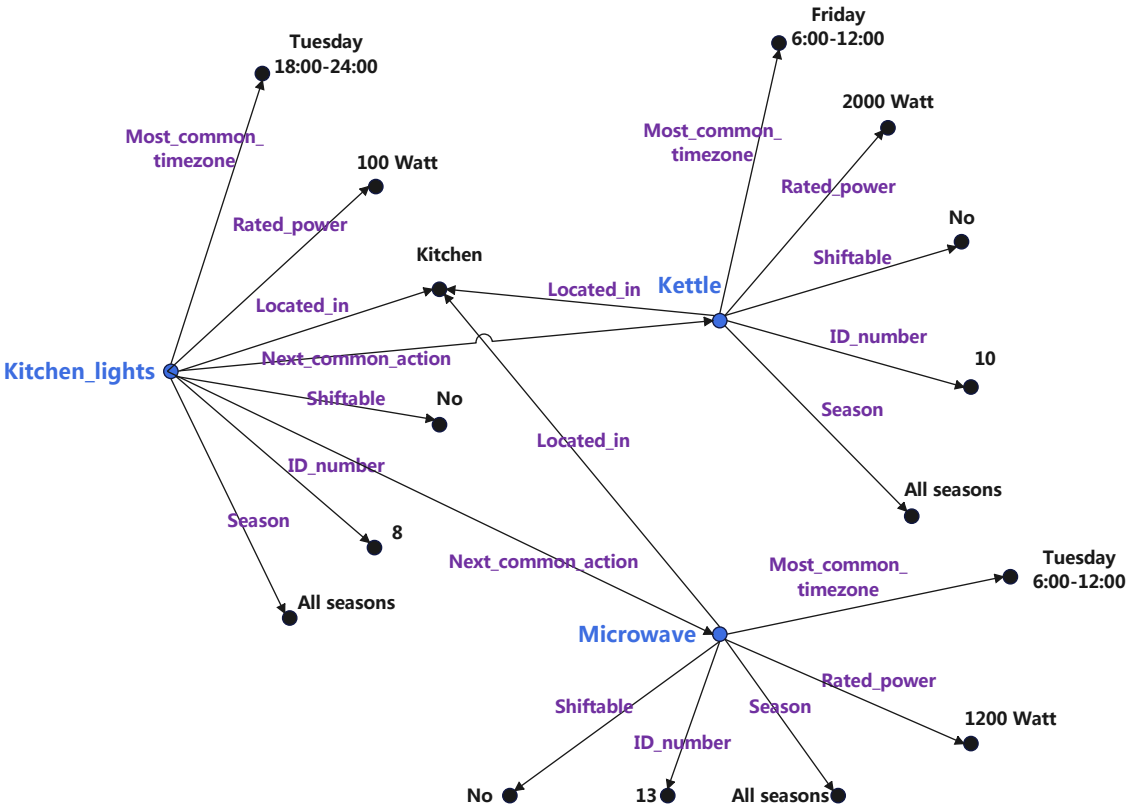

Figure 3.5: Constructed domestic appliances knowledge graph. Each household appliance has attributes such as an ID, shiftability, rated power, usage time, and appliance location. If there is a sequential order of usage between two appliances, a line is established between them.

kitchen lights, kettle, and microwave are all in the kitchen, so they both point to the 'kitchen' point. Actions commonly taken after turning on the kitchen light include using the kettle or microwave, thus establishing relationships (edges on the graph) between appliances.

Additionally, there are hidden relationships, such as the frequent use of kitchen lights and microwaves on Tuesdays, indicating a hidden relationship between them. Therefore, more potential relationships can be discovered by knowledge graph algorithms. Information about appliance demand response is also added to the graph; for example, all three appliances in the diagram are non-shiftable. Demand response will not significantly impact user's expected satisfaction if an appliance is shiftable, like a washing machine or dishwasher. However, for non-shiftable appliances, demand response has a more significant impact on user's expected satisfaction. Therefore, the information provided in the diagram will also influence subsequent demand

response algorithms in the optimization module.

3.6.2 Recommendation system parameters

For Embeddings, 37 out of 54 kinds of domestic appliances (which is shown in table 3.1), 9 kinds of locations (including the kitchen, living room, office, bedroom, utility room, children's room and 3 undefined locations), 28 kinds of time periods and 3 kinds of relationships (including "used before/after", "used during the time period" and "located in the room") are generated as embeddings, as discussed in the section 3.1.1 and 3.2. The dimension is set as 128 for all embeddings.

Table 3.1: Parameters for the recommendation system

Parameter in recommendation system	**Values**
Device Embedding dimension	128
Relation Embedding dimension	128
Time Embedding dimension	128
Embedding input nodes	3
KGAT input nodes	384
(Comparison) RNN input nodes	384
(Comparison) CNN input nodes	384
(Comparison) DNN input nodes	384
DNN hidden nodes	384×128
DNN output nodes	1
Tearning rates	0.001
Training rounds	30
Training iterations in a round	50
Mini-batch size	1024
Optimizer	Adam
Loss function	Mean-Squared-Error
Activation function	sigmoid and tanh

For the proposed KGAT in DARK and traditional KGAT, the input size is $128 \times 3 = 384$, and the output size is also 128. In the embedding input size, 3 means device A, B, and time. After the KGAT layers, a DNN layer is set as the classifier. To evaluate the effectiveness of the KGAT algorithm, other neural network structures, including DNN layers, CNN layers and RNN layers, were implemented as alternatives to the KGAT knowledge graph layer, all of which have the dimension

of 384 × 128. In addition, the original KGAT algorithm without modifications was also included in the comparative simulation. The detailed parameter is shown in table 3.1.

3.6.3 Recommendation system accuracy measurement

This project defines the final recommended results as the top three most likely appliances the recommendation algorithm generates. Precision is determined by comparing with the labels and serves as an accurate standard for measuring the accuracy of the recommendation system. If there is an appliance among these labelled appliances (meaning it is the appliance to be recommended), the recommendation is considered accurate; otherwise, it is considered inaccurate. The reason for selecting the top three recommendations is that sometimes, the recommendation system tends to suggest using multiple appliances with a higher probability.

The results of the recommendation system are values ranging from 0 to 1, with a value closer to 1 indicating a higher likelihood of the next action being triggered.

3.7 Results

This section presents the simulation results of DARK, including the recommendation system's recommendation outcomes, the recommendations' precision, the interpretability, and the energy optimization results derived from the recommendations.

3.7.1 Results for algorithm performance

This section analyzes the algorithm's performance from various perspectives, including recommendation results, accuracy and space complexity.

Recommendation results

The complete recommendation list is shown in Fig. 3.6. In the image, the length of the horizontal bars represents the recommendation results, and the vertical axis represents various types of electrical appliances. The image shows the recommendation ranking of 36 types of electrical appliances, ranging from the highest-scoring

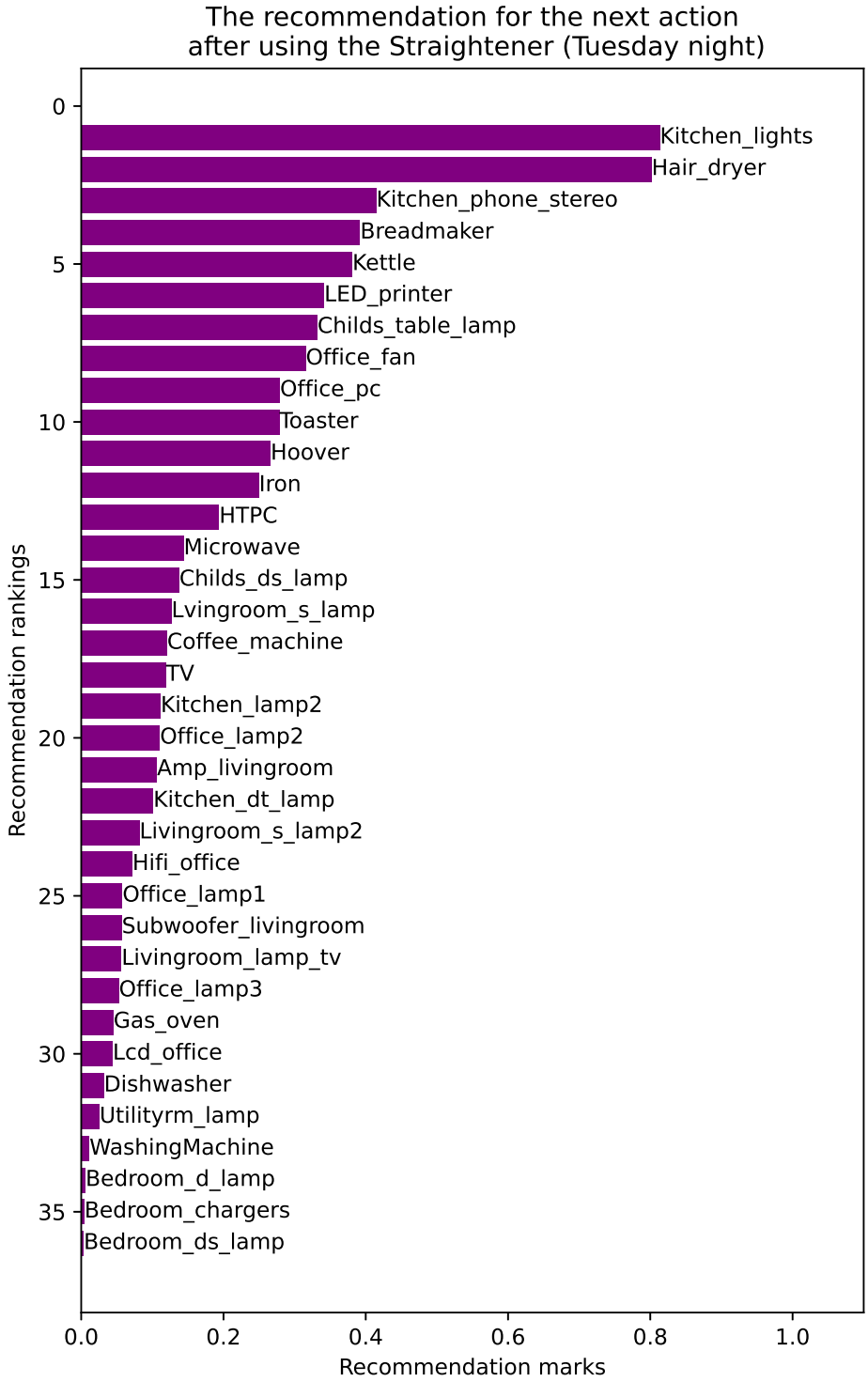

Figure 3.6: Recommendation results for the straightener. On Tuesday nights, after using the Straightener, the recommendation system determines the next two highest actions, which are using the kitchen lights and hairdryer, with its recommendation far exceeding other appliances.

'kitchen lights' at the beginning to the 'bedroom lamp', which the recommendation system deems least likely to be used. After using the straightener on Tuesday night, the three most likely appliances to be used are the kitchen light (0.81), the hairdryer (0.80), and the kitchen stereo (0.41). It is worth noticing that the first two results are significantly better than the third and beyond, demonstrating the good quality of distinct emphasis of the recommendation results.

The graph does not differentiate the reasons for the recommendations. Still, it's easy to conjecture that the recommendation of 'kitchen lights' is because the recommendation time is Tuesday evening when lights are generally needed. Therefore, the reason is that they are often used simultaneously. Regarding the 'hair dryer' recommendation, it is because it is often used consecutively with the 'straightener'. Hence, their recommendation reason is presumed to be their frequent consecutive usage. Further interpretation about these recommendations will be elaborated in the subsequent simulations.

Accuracy and converges situation

Fig. 3.7 shows a selected instance where all other conditions were the same, representing typical algorithm performances. It can be observed that DNN and CNN yielded notably inferior results. However, they do show training effectiveness. Assuming the selection of 3 devices at random from a set of 37 devices, the probability of including a specific number is $\frac{c(36,2)}{c(37,3)}$, which is around 8.1%, and c means combination calculation in mathematics. The recommendation accuracy of DNN and CNN reaches approximately 30%. The performance of the RNN algorithm steadily improved over time, but the improved KGAT achieved the highest accuracy (averaged accuracy: 93.4%) in the first round and provided the most accurate recommendations, which averaged 7% accuracy improved compared with the traditional KGAT (averaged accuracy: 87.3%). It can be seen that the proposed KGAT has learned the latent patterns of domestic appliance recommendations, achieving the highest precision in recommendations, and KGAT algorithms have a greater advantage in the early stages of the recommendation process.

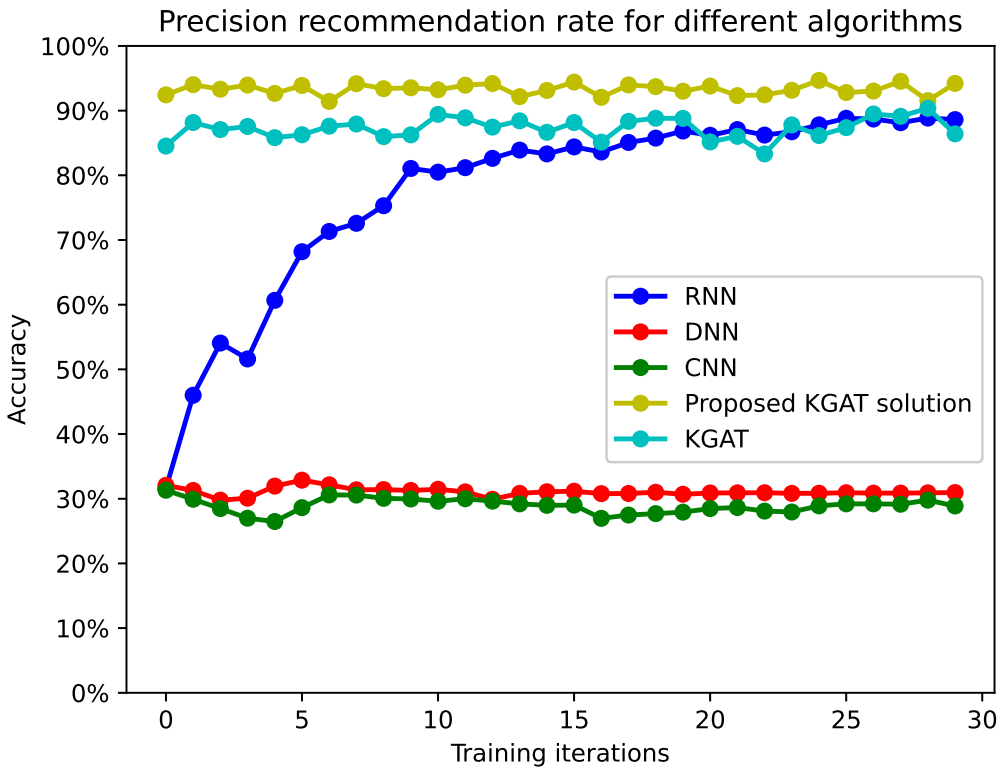

Figure 3.7: Recommendation accuracy: the proposed KGAT achieved the highest accuracy among all the network structures.

3.7.2 Results for interpretations

This section's recommended results have the same format as shown in Fig. 3.6, but the analysis is focused on the top 15 recommendations as they are more important. The interpretation of the recommended results utilizes different colours: blue indicates a sequential usage order of two appliances, green indicates frequent usage of two appliances during a specific time period, and yellow indicates a positional dependency between two appliances. Three examples are provided in this section to illustrate the varied interpretations of the recommended results by the recommendation system.

Reason-Sequence

This reason means that 2 appliances are always used together. The recommended score on the horizontal axis is the inner product of embeddings from the recommen-

dation algorithm. After using the hairdryer, the system's best-recommended result is the straightener, with 72% possibility of using them in sequence and the rest 28% of the same time for interpretation. The probability analysis is shown in Fig. 3.8. Since the recommendation score for the 'straightener' is significantly higher than that of other appliances in this context, it is feasible to disregard the other recommendation results, the recommendation result is determined with a relatively high probability.

This interpretation aligns with common sense, as these two appliances are often used together after showering. It is worth noting that the results obtained here correspond with those obtained in Fig. 3.6. This further demonstrates the stability of the proposed DARK algorithm.

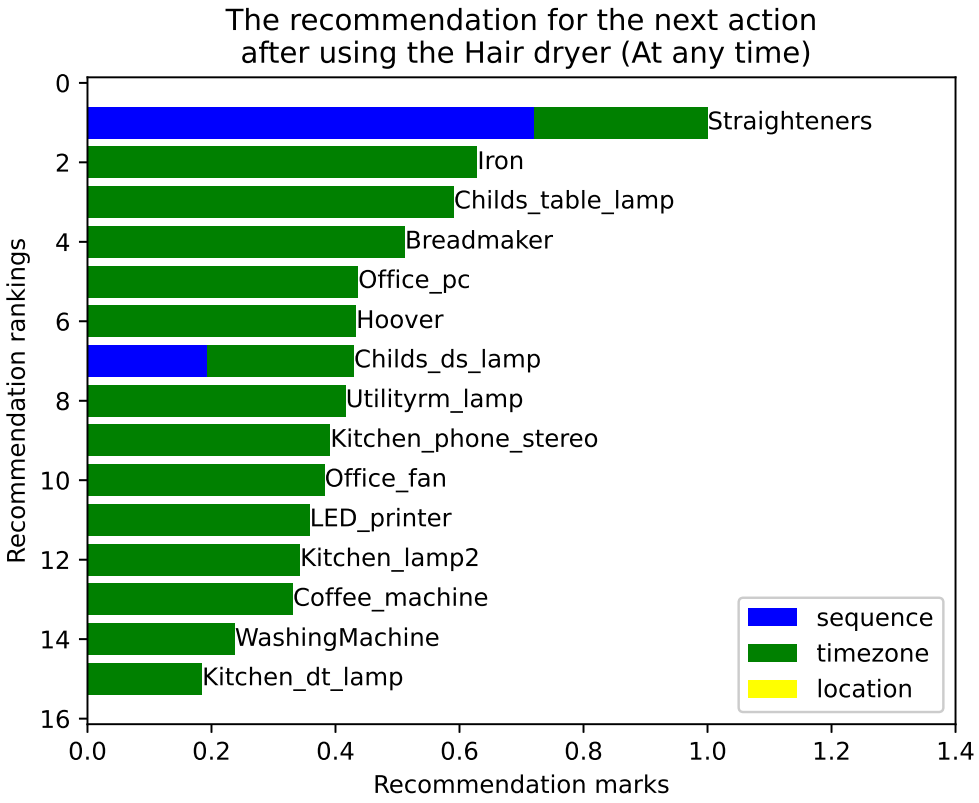

Figure 3.8: The recommendation reason is that the appliances are used in sequence. After using a hair dryer, the recommendation system predicts that the next most likely action is using straighteners. The reason for this is that straighteners are often used sequentially after hair dryers.

Reason-Timezone

The second reason is that these two appliances are frequently used during the same time period, as shown in Fig. 3.9, on Sunday nights, lights, TV, and subwoofer are also recommended with more than 90% reason for using in the same time because they are always used on Sunday night. Also, in Fig. 3.10, on Tuesday nights, according to the resident's habits, appliances such as lights, office lamps and appliances for entertainment such as television, Hifi and HPTC are recommended.

These recommendations also align with the common sense. This recommendation result also reflects that multiple high-probability recommendations can exist when the reasoning for recommendations is primarily based on the timezone. This is because many appliances are used simultaneously during a specific period, and there are no other significant reasons to create a substantial difference in the recommendations between these appliances. Additionally, the timezone is the most common reason in this recommendation system, as two appliances are most likely to be used at the same time.

Reason-Location

This reason means both appliances are in the same location. As is shown in Fig. 3.10, when using the Gas Oven on Tuesday afternoon, the recommended appliance list includes the kettle, microwave, and other kitchen lights, with the reason of 40%, 42% and 49% for using them in the same location respectively. This also aligns with common sense, as using appliances for cooking in the evening often involves turning on lights or using a kettle to boil water in the same location.

The most possible recommendation result for each appliance

Fig. 3.11 illustrates the recommendation results for different appliances. On the left are the target appliances for recommendations, and on the right are the appliances most likely to be used next after using the target appliance in all instances. The results are sorted by the degree of recommendation from 0 to 1. The recommended appliances can be roughly categorized into several groups:

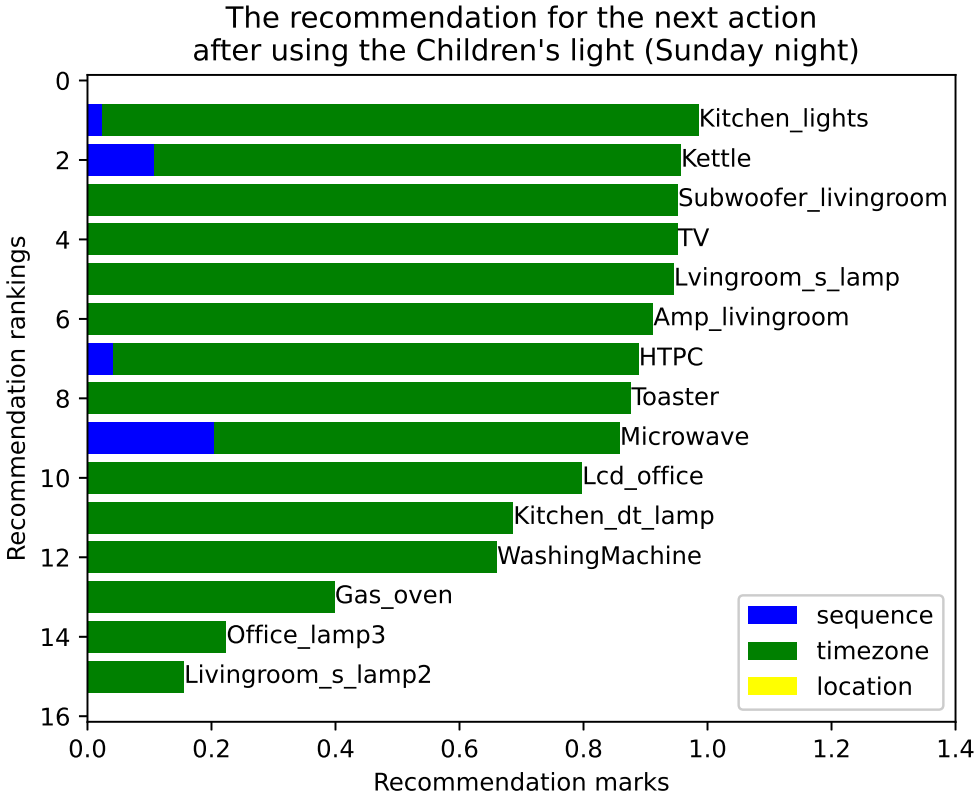

Figure 3.9: The recommendation reason is that the appliances are used in the same time zone. After using the children's lights on Sunday evening, the most commonly used lights are the kitchen lights, as they are often used at the same time as the children's lights.

1. Living Room Series: Such as the television 'TV', lamp for TV in the living room 'Livingroom_lamp_tv', Amplifier in the living room 'Amp_livingroom', 'HTPC', and subwoofer 'Subwoofer_livingroom' are ranked as the top recommendations for each other, which align with daily usage logic, as using the TV often requires auxiliary devices like a lamp or speaker.

2. Kitchen Series: Including lights 'Kitchen_lights' and 'Kitchen_dt_lamp', 'Kettle', 'Toaster', 'Gas_oven', 'Coffee_machine', and kitchen phone stereo 'Kitchen phone_stereo'. These appliances are recommended as the top choices for each other, which makes sense since they often collaborate. For example, cooking may involve turning on the lights.

3. Bedroom Series Such as Bedroom's lamp 'Bedroom_d_lamp' and Bedroom's charger 'Bedroom_chargers' are recommended as top choices for each other,

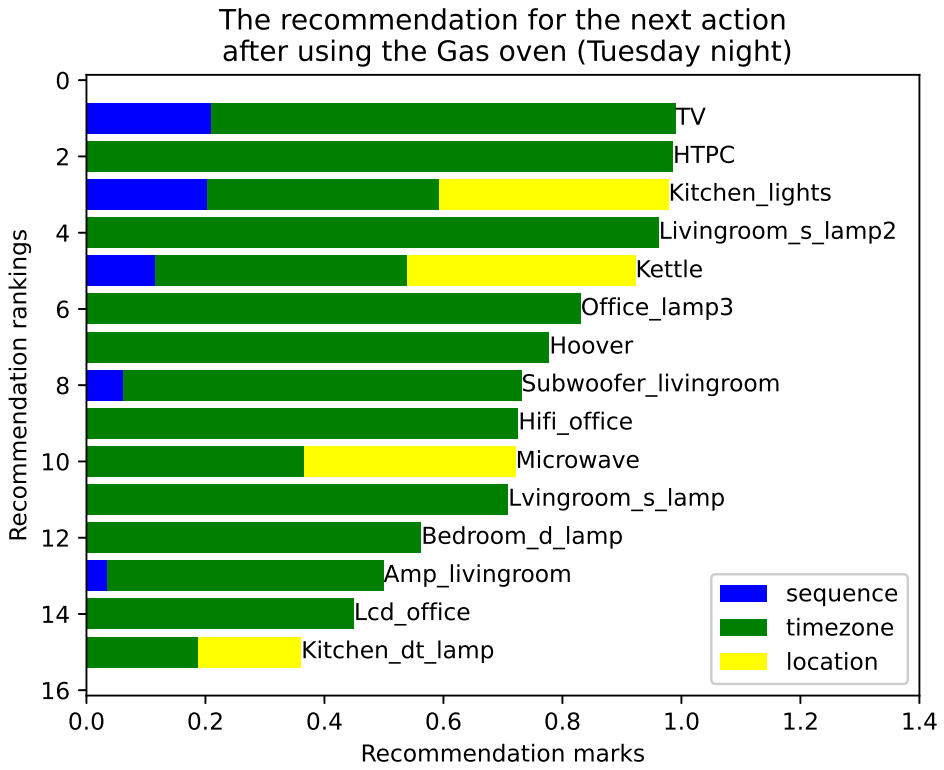

Figure 3.10: The recommendation reason is that the appliances are in the same location. On Tuesday evenings, after using a gas oven, the recommendation system predicts that the next three most likely actions are turning on the TV, using the HPTC, and turning on the kitchen lights. Regarding the kitchen lights, one reason for this prediction is their location-based association.

with the primary reason being location, aligning with typical usage patterns.

4. Office Series: Such as Office lamps 'Office_lamp1', 'Office_lamp2' and 'Office_lamp3', LCD device 'Lcd_office', and Hifi device 'Hifi_office' are recommended as top choices for each other.

5. Other recommendations: Appliances like 'hair_dryer' and 'straightener' are often used with each other because of the use sequence. A similar result is 'Utilityrm_lamp', and washing machine 'WashingMachine' are recommended because they have always been used at the same time. Another case is the Children's Lamp 'Childs_table_lamp' and 'Childs_ds_lamp' are recommended to work with each other.

6. Example of failed recommendations: The last six items are considered un-

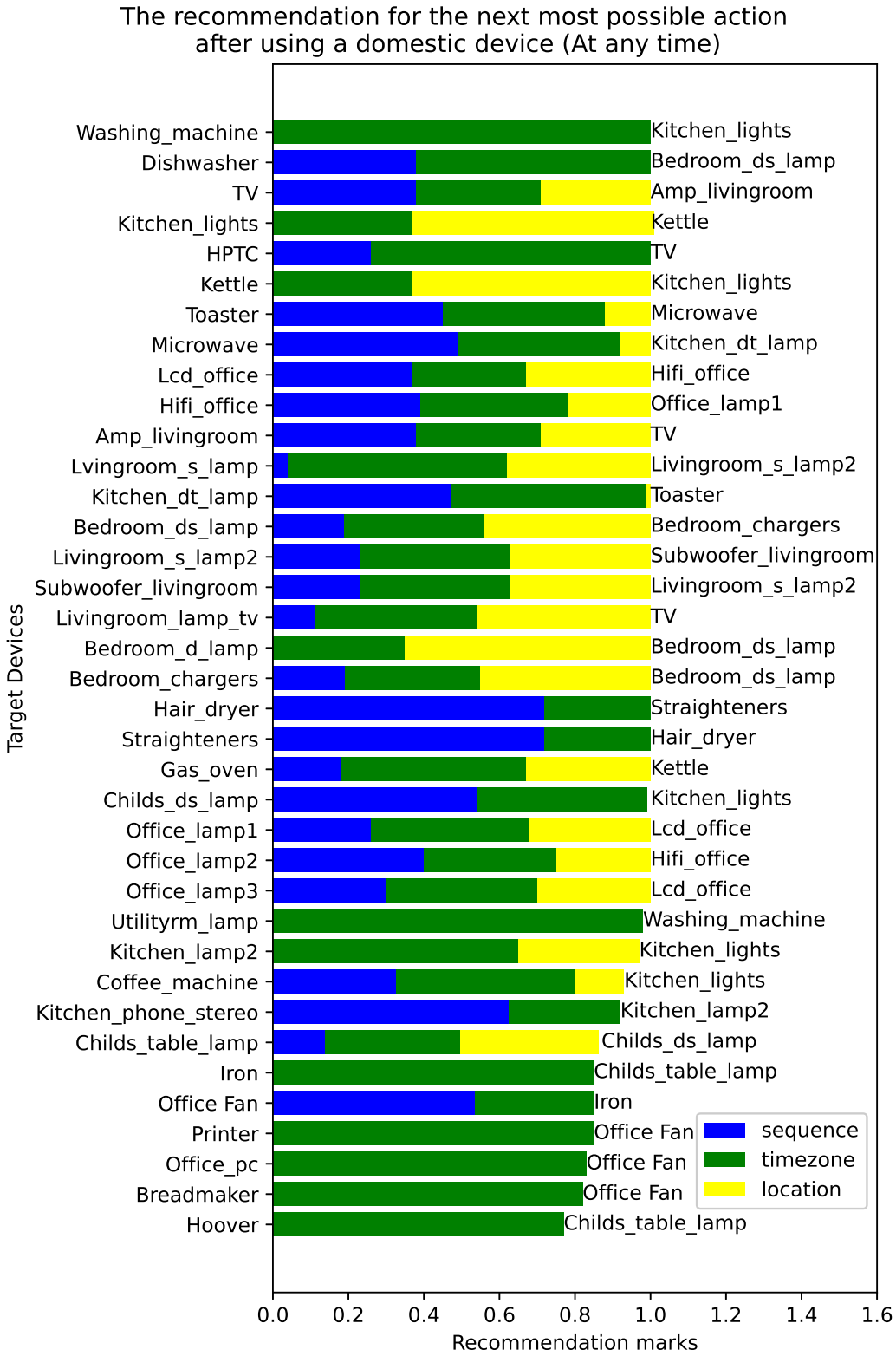

Figure 3.11: The most possible recommendation result for each domestic appliance, the left side represents appliances that have already been used, and the right side shows the appliances that are most likely to be used next.

successful recommendation results, as they did not recommend an appliance with high probability. For appliances like the 'Hoover', 'Printer', and 'Office fan', inaccurate recommendation results may occur due to a lack of data or unrepresentative data.

It can be observed that the proposed system effectively captures relationships between appliances and provides reasonable interpretation for the recommendations.

3.7.3 Algorithm space complexity

Algorithm space complexity is a metric that measures the amount of memory space required by an algorithm or data structure [188]. Particularly in the context of large network architectures, understanding space complexity aids in estimating computational costs and planning for hardware resources. It also serves as a metric for assessing algorithms' scalability in structures, which is meaningful for selecting edge computing devices in a home setting. This chapter summarized the space complexity for the DARK algorithm.

1) Embeddings: The space complexity of the embedding layer is $128 \times N$, where N is the kind of embedding. In this simulation, 9 kinds of locations (with 3 unknown locations), 28 kinds of time periods, 37 kinds of domestic devices and 3 kinds of relations. The space complexity of embeddings is $128 \times (9+28+37+3) = 9856$.

2) KGAT layer: In KGAT, all operations are based on the computation and aggregation of embeddings. Weight matrices are introduced in both attention calculation SP_{AC} and aggregation operations SP_{AO},

$$SP_{\mathrm{KGAT}} = \sum_{i=1}^{I} (SP_{\mathrm{AC}_i} + SP_{\mathrm{AO}_i}) \ . \tag{3.11}$$

In the attention calculation, the space complexity is $SP_{\mathrm{AC}} = d_E \times d_R$, where d_E is the dimension of embedding and d_R is the dimension of relation. In the aggregation operation, the space complexity is $SP_{\mathrm{AO}} = d_E \times d_E$. The sum of these two contributes to the overall space complexity. For example, if the dimension of the device d_E and d_R relation are all 128, then the space complexity is $128 \times 128 \times 2 = 32768$, where 2 means SP_{AC} and SP_{AO} are all 128×128.

3) Proposed KGAT layer: In addition to the KGAT layer, due to the need for sampling for each category, distribution information of the graph data Graph_indexes needs to be recorded,

$$SP_{\text{Proposed KGAT}} = SP_{\text{KGAT}} + \text{Graph\_indexes} \ . \tag{3.12}$$

For example, the distribution of 7 locations, 28 timezones, and 37 device types. Although this introduces additional storage, the data for distribution information is simple and only adds minimal storage burden. So in this simulation, the space complexity is 7+(37 × 28)+(37 × 36)+32768=35143, where 7 is the number of different locations, 37 × 28 is the number of devices used in a different time, 37 × 36 is the number of devices used after another device, and 32768 is from 2).

4) DNN layer: A DNN classifier of size 384 ×1 was added in the simulation. The space complexity of a DNN refers to the size of its weights and biases, formulated as

$$SP_{\text{DNN}} = \sum_{i=1}^{I} (W_i + B_i) \ , \tag{3.13}$$

where W_i refers to the weights for DNN and B_i refers to the biases. For example, a DNN with a size of 384 × 1 has a size of 384 × 1 + 1 = 385, where 384 is the size of the input layer, the first '1' is the size of the output layer, and the second '1' is the size of the bias.

In conclusion, the proposed KGAT algorithm's space complexity growth mainly depends on the dimensionality of embeddings. The attention calculation and aggregation operations increase quadratically with the embedding dimension. The total space complexity for traditional KGAT is 9856+32768+385=43009, and the space complexity for proposed KGAT is 9856+35143+385=45384. The proposed KGAT only introduces a minimal increase (around 5.5%) in space complexity compared to traditional KGAT algorithms.

Comparison with other interpretable algorithm

Although DNN, CNN, and RNN can all utilize explainable machine learning methods [150], [131], [132], such as LIME and SHAP [189] [169], to explain the reasoning

behind their predictions and recommendations, the interpretability of black-box models is far inferior to that of interpretable structures based on graphs and trees. The proposed KGAT algorithm's graph-based structure allows it to do reasoning about recommendation results, achieving accurate traceability as a white-box model. Therefore, the proposed algorithm is more suitable for scenarios related to human activities, as any erroneous decisions can be traced and improved.

Apart from that, during the embedding aggregation step in almost all knowledge graph algorithms, calculations are made to determine the weight with which the embedding of a neighbouring tail node influences the head node. Hence, training knowledge graphs are also inherently interpretable. Consequently, KGAT and its traditional counterpart offer better interpretability than DNNs, RNNs, and CNNs.

3.7.4 Results for demand response

It is assumed that the rated power of all appliances is adjustable within 0.8 - 1.2 times. To simplify the problem, the assumption does not consider the appliances' usage duration and real consumption. Only the adjusted power (unit:wh) is taken into account. In the simulation, it is assumed that there are 3 kinds of satisfaction coefficient in table 3.2: The first kind is adjustable appliances, including washing machines and dishwashers, the parameter df_i in (3.9) is set as 0.0001, which means it will produce less dissatisfaction if the power is adjusted. The second kind is mainly entertainment appliances, including TV, HTPC, etc. and the parameter df_i in (3.9) is set as 0.0003. The third kind is essential household appliances, including microwaves, gas ovens, office PCs, etc. The parameter df_i in (3.9) is set as 0.0005, which means it will cause dissatisfaction if the demand response adjusts it. The value of df_i is manually set based on life experience. The values 0.0001, 0.0003 and 0.0005 are chosen because they can normalize the dissatisfaction score to a range of 0-400. After optimization by the Generic algorithm with the objective dissatisfaction calculated in (3.9) and the averaged power calculated in (3.10), the Pareto front generated by the optimization algorithm is shown in Fig. 3.12, where each point represents a set of device power settings. The detailed dissatisfaction coefficient and df_i rated power are attached in Table I. Please note that the dissatisfaction level in

Table 3.2: Rated power and user's expected satisfaction coefficient of home appliances

Device	Rated power(W)	Satisfaction coefficient df_i
Washing machine	1700	0.0001
Dishwasher	2350	0.0001
TV	130	0.0003
Kitchen lights	150	0.0003
HTPC	70	0.0003
Kettle	2400	0.0005
Toaster	1580	0.0005
Microwave	1510	0.0005
LCD office monitor	50	0.0003
Hi-Fi office stereo	15	0.0003
Breadmaker	580	0.0005
Living room amp	45	0.0003
Living room floor lamp	1100	0.0003
Hoover (vacuum cleaner)	2000	0.0001
Kitchen desktop lamp	40	0.0003
Bedroom desk lamp	80	0.0003
Living room side lamp	20	0.0003
Living room subwoofer	50	0.0003
Living room TV cabinet lamp	25	0.0003
Kitchen table lamp	20	0.0003
Kitchen phone stereo	20	0.0003
Utility room lamp	45	0.0003
Bedroom table lamp	60	0.0003
Coffee machine	1270	0.0003
Bedroom chargers	30	0.0003
Hair dryer	1680	0.0003
Straighteners	500	0.0003
Iron	1800	0.0003
Gas oven	60	0.0005
Child's table lamp	15	0.0003
Child's desk lamp	50	0.0003
Office desk lamp 1	30	0.0003
Office desk lamp 2	25	0.0003
Office desk lamp 3	20	0.0003
Office PC	240	0.0005
Office fan	50	0.0003
LED printer	900	0.0005

Pareto front $f_{\text{dissatisfaction}} = -f_{\text{satisfaction}}$ in (3.9).

In Fig. 3.12, the blue points represent the optimized Pareto fronts, which have

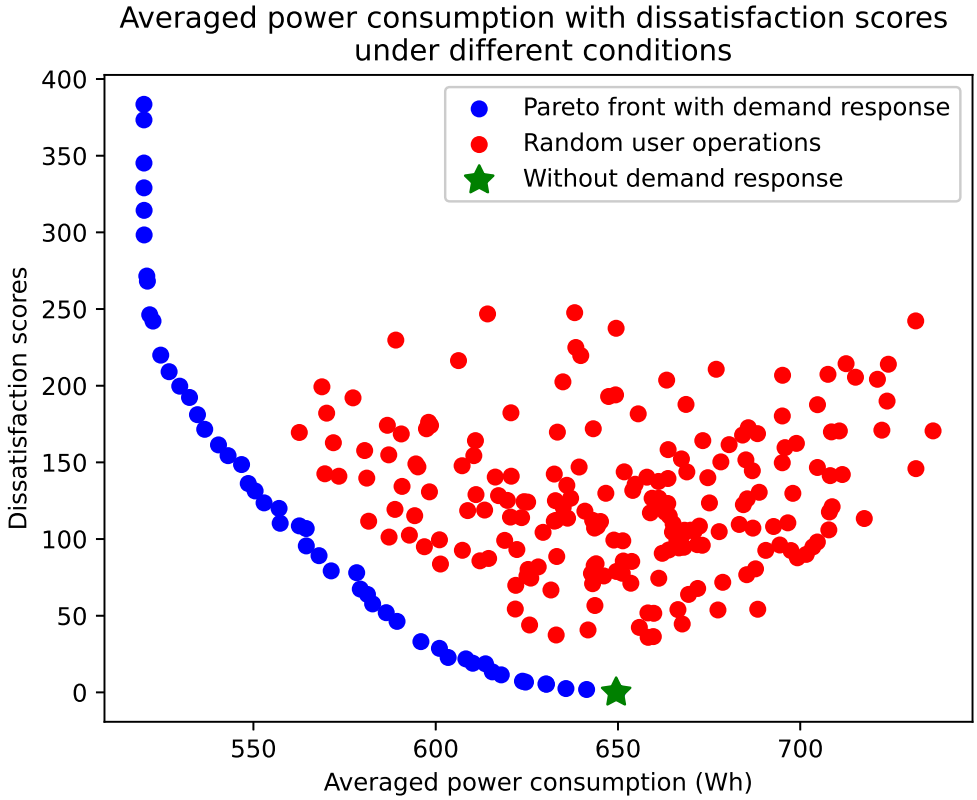

Figure 3.12: The Pareto front for demand response optimization, the optimized Pareto points (marked in blue) significantly outperform the unoptimized points (marked in red) in terms of both user's expected satisfaction and average power consumption.

the most efficient energy consumption and user's expected satisfaction balance. The red points represent the values obtained by simulating user operations, assuming that the power range of each electrical appliance is a random multiple of 0.8 to 1.2 times the rated power, calculating the energy consumption and dissatisfaction level (averaged Power between 560 and 740 watts). It can be seen that the optimized Pareto front blue points can achieve lower dissatisfaction levels than the red points while saving energy consumption. The green star dots represent rated appliances' power (averaged Power = 650 watts and user's expected dissatisfaction level= 0). It can be seen that compared to the green dots, the optimized power can be reduced by up to (650 watt-520 watt)/650 watt=20% with the proposed algorithm (the Averaged Power of the rightmost blue point is 520 watt).

According to the user's preference, the system can automatically assign power

settings to each electrical appliance based on the Pareto fronts to determine the choice that leans towards a higher dissatisfaction level or energy-saving.

3.8 Conclusions

This chapter has established a recommendation framework, DARK, that accurately suggests and optimizes the user's next actions. The key contributions of this chapter include:

1) The appliance information from the existing UK-DALE dataset was modelled as a knowledge graph to predict the user's next possible actions. The constructed graph effectively assists the household device action recommendation system, aiding the algorithm in uncovering implicit information.

2) The KGAT algorithm was used and improved, enhancing the KGAT's performance in the proposed DARK framework.

3) Interpretation methods for the recommended results were designed and implemented. In the proposed system, reasons for recommendations are categorized into three types: frequently sequential used, simultaneously used, and used in the same location. Percentages are assigned to each reason based on the design principles of embedding.

4) Demand response optimization was carried out based on the expected recommended actions, while considering energy consumption and residential satisfaction. This optimization resulted in different power level settings for each appliance. By optimizing the expected outcomes of the recommendations, a Pareto front is achieved, balancing both expected residential satisfaction and average power consumption.

For the recommendation module, it can be observed from the simulation that the proposed system can provide recommendations with higher accuracy; the proposed recommendation system can effectively predict the next action with an accuracy of up to 93.4%. This result significantly outperforms CNN and DNN layers, with convergence speeds and accuracy surpassing the RNN layer. This improved KGAT layer shows an approximate 7% increase in effectiveness when compared to the unmodified version.

For the interpretation module, an interpretation based on embedding has been found and used in the recommendation system. Compared to the black-box explainable learning algorithms, the proposed algorithm can trace the reasons for the recommendations through the graph. The proposed method can determine the percentage contribution of each path (interpretation reason), thereby enabling users or energy managers to understand the reasons behind the recommendations better.

For the optimization module, based on the recommendation's result, the proposed system can adjust the appliances' power based on their expected satisfaction level and energy consumption. The optimized power can be reduced by up to 20% while maintaining the users' satisfaction level.

This chapter focuses on domestic appliances' recommendation, interpretation and power optimization. For further energy and carbon optimization, the renewable energy and energy storage systems in the home microgrid need to be considered. The details will be discussed in the next chapter.

Additionally, this chapter utilizes household electricity data. Although federated learning can be used to keep the data locally, the gradients inevitably contain some private information. Chapter 5 aims to investigate this issue.

CHAPTER 4

Energy and carbon emission scheduling system

Due to the concern of fossil fuel depletion, integrating renewable and distributed energy sources in power grids is needed. The microgrid is a promising integration solution due to its potential to improve grid operation efficiency, which has advantages such as integrating renewable energy, increasing flexibility and reliability, reducing transmission losses and increasing energy independence [190]. A microgrid can operate independently of the primary power grid or be connected to it, which can be managed through advanced control systems to meet local energy demands. This also brings more possibilities for energy scheduling optimization.

Unlike the single household device action recommendation system in Chapter 3, this chapter focuses on the home microgrid systems of several households, expanding the scope from a single home to that of a microgrid. The home microgrid is a small-scale microgrid consisting of several households [191] [192]. It refers to a distributed power system based on small-scale renewable energy sources (such as solar panels, wind turbines, etc.) and energy storage systems (such as batteries). On the one hand, in the home microgrid, privacy issues have become more prominent [193]. On the other hand, improving the operational efficiency of home microgrids, realizing low carbon emissions, lowering electricity costs, and enhancing renewable energy

penetration has been a challenge. To achieve these objectives, this chapter aims to build a scheduling algorithm using deep reinforcement learning and federated learning, resulting in high energy efficiency while protecting privacy.

Another objective is that in a bi-objective scheduling model, sometimes it is necessary to set biases for the optimization objectives. For example, optimizing the accumulated carbon emissions can be more important than optimizing electricity costs, and vice versa. Making the scheduling model adjustable is an interesting research direction for this chapter.

The proposed home microgrid system consists of 4 homes. Each home has 2 sub-systems, i.e., the supply sub-system and the load sub-system. The supply sub-system consists of a Photovoltaic (PV) panel, batteries, a power converter and the power grid, where the power converter integrates various energy sources into households. The load sub-system contains home loads. The overall structure is shown in Fig. 4.1.

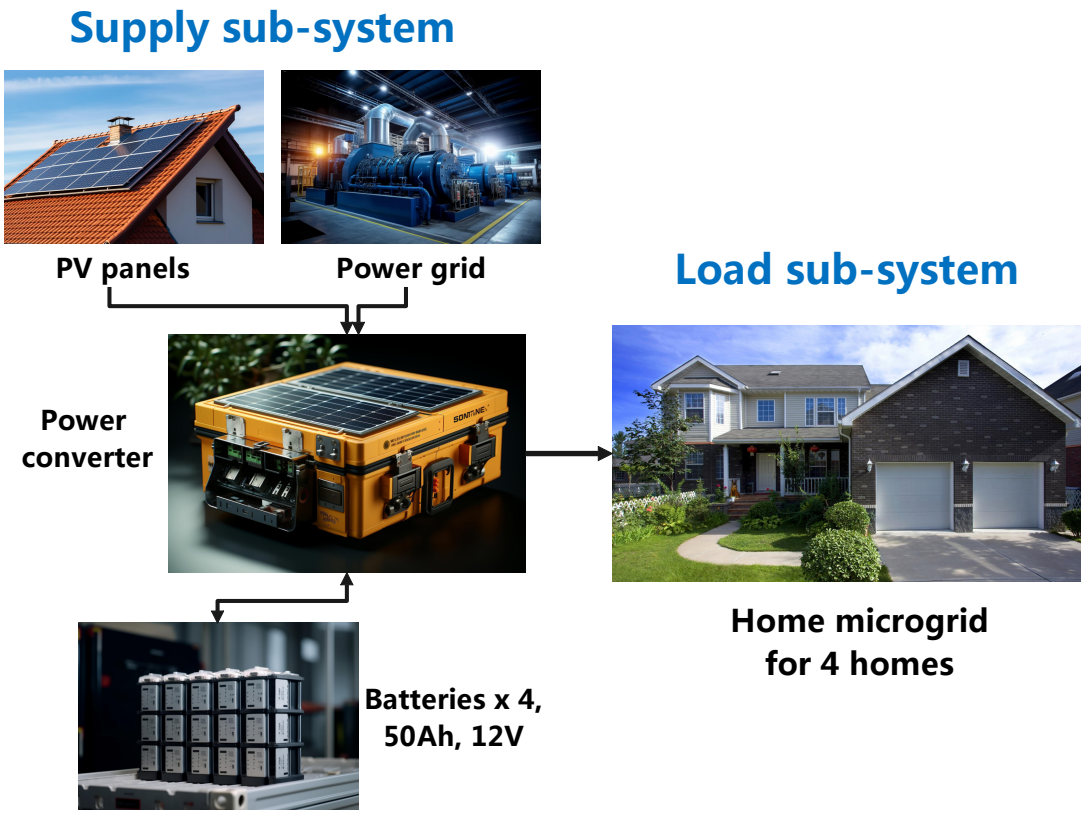

Figure 4.1: The proposed system architecture with PV panels, a power grid, battery groups and a power converter.

The PV panel is the power source when the solar irradiance is enough. The

rechargeable battery stores energy whenever there exists a PV energy surplus. The battery can be discharged to supply home loads if there is no or low solar irradiance. When the battery capacity is low, the power grid is the main power source to fulfil the home load.

Figure 4.2 shows different modules' relationships. The proposed framework can be divided into the following four modules.

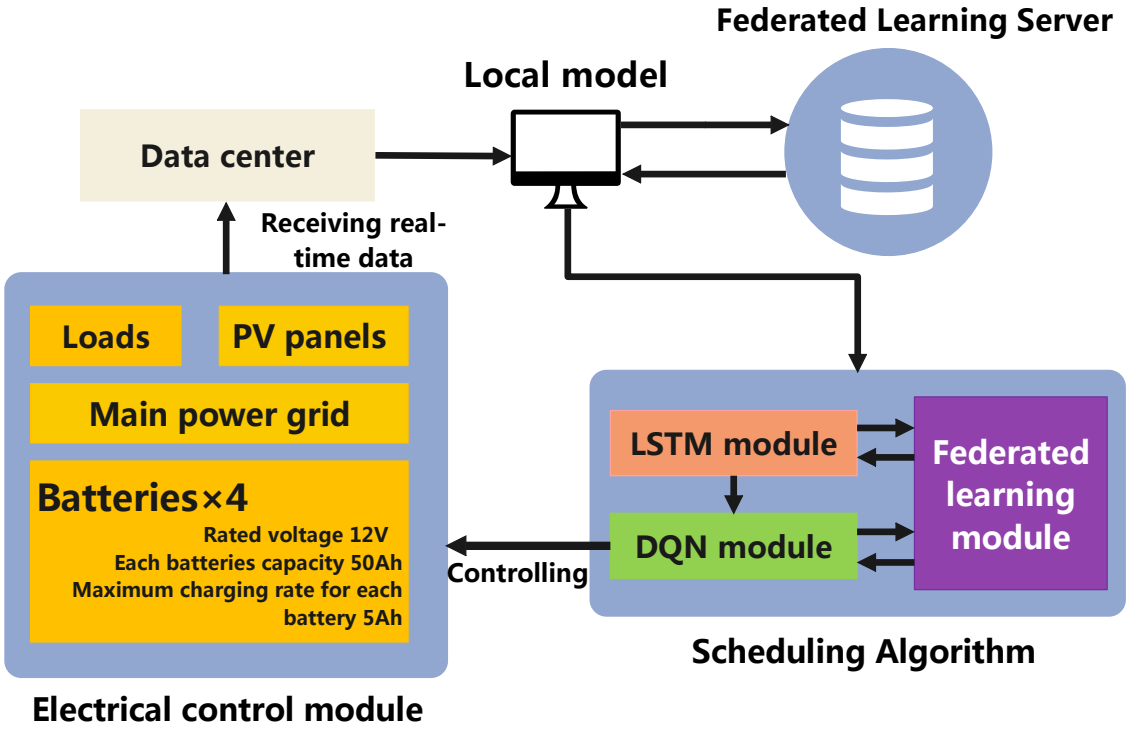

Figure 4.2: The proposed algorithm architecture with LSTM, DQN, federated learning and electrical controlling modules.

- The data module (the white section) is used for data collection and allocation. The load data and PV data are used for electrical central control, and the electricity price and carbon emission data are sent to the DQN module for scheduling. All the data is also sent to the LSTM module for many types of predictions [137].

- The electrical control module (the orange sections) is used to evaluate the electricity costs, the carbon emissions and the remaining battery energy (every half an hour) with the states of the PV panels and batteries. These are based on the information from all the other modules.

- The LSTM module (the light red section) is used to predict solar irradiance, loads, carbon emissions and electricity prices in the next future half an hour using the LSTM network [133]. The LSTM module mainly collaborates with the DQN module because the DQN operations environment needs one more prediction time step. It also collaborates with the federated learning module for privacy protection.

- The DQN module (the green sections) determines when and how much electricity should be purchased according to the real and the predicted data in the next half an hour to minimize carbon emissions and electricity costs.

- The federated learning module (the purple section) is used to collaborate with the LSTM module and the DQN module using different homes' data, providing model aggregations, breaking the data isolation and protecting privacy.

4.1 Electrical controlling module

In this section, the background models, including the photovoltaic model, battery model, load model and scheduling model, will be introduced.

4.1.1 Photovoltaic model

PV energy generation can be modelled by (4.1) [35],

$$E_{\mathrm{PV}}(n) = A \times SR(n) \times \eta_{\mathrm{PV}} \times k_n \ , \tag{4.1}$$

where $E_{\mathrm{PV}}(n)$ is the energy generated from PV (Wh). A is the effective contact area (m^2). $SR(n)$ is the averaged solar irradiance ($\mathrm{W/m}^2$), n refers to the n-th time interval. η_{PV} is the solar-electric energy conversion efficiency. k_n is the length of the time interval. The constraints are shown in

$$A{>}0, SR(n) \geq 0, 1{>}\eta_{\mathrm{PV}}{>}0, k_n{>}0 \ . \tag{4.2}$$

which means the effective contact area, solar irradiance, time and efficiency cannot be below zero.

4.1.2 Rules for energy supply and batteries

The battery model can be given by (4.3) and (4.4) [194].

$$E_B(n) = E_B(n-1) \times (1-\eta_s) + (E_{ch}(n) - \frac{E_{dis}(n)}{\eta_c})\eta_b \ , \tag{4.3}$$

$$SoC(n) = E_B(n)/(C_B \times V_{Ra}) \ , \tag{4.4}$$

where $E_B(n)$ is the battery energy (Wh), η_s is the charging efficiency, $E_{ch}(n)$ is the charging energy (Wh), η_c is the inverter efficiency. η_b is the battery efficiency. $E_{dis}(n)$ is the discharging energy for the load (Wh). $SoC(n)$ is the state of charge, C_B is the maximum battery capacity (Ah) and V_{Ra} is the battery voltage (V). The constraints are shown in (4.5), (4.6) and (4.7).

$$E_{ch}(n) = E_{\mathrm{PV}}(n) + E_{cb}(n) \ , \tag{4.5}$$

$$SoC(n) \geq 0, E_{dis}(n) \geq 0, E_{ch}(n) \geq 0 \ , \tag{4.6}$$

$$SoC(n) \leq SoC_{\mathrm{lim}}, E_{dis}(n) \leq E_{\mathrm{maxd}}, E_{ch}(n) \leq E_{\mathrm{max}}, 1{>}\eta_s, \eta_c, \eta_b{>}0 \ , \tag{4.7}$$

where $E_{cb}(n)$ is the amount of energy bought from the main power grid to charge the battery. Please note that there are 2 kinds of energy from the main power grid, one kind is the power to charge the battery $E_{cb}(n)$, the other is the energy used to power the load, the total energy bought from the main power grid is $E_{i,\mathrm{bought}}$ from formula 4.16. The state of charge, the charging energy and the discharging energy cannot be below zero. SoC_{lim}, E_{max} and E_{maxd} are the limitations of the battery, the maximum energy charging and discharging energy in a time interval, respectively.

The battery can be charged using either PV or the main power grid. Grid energy can be purchased at any time and stored in the battery. Electricity will not be bought from the power grid when the battery energy is sufficient (SoC $>$ 90%). When the battery energy is insufficient (SoC $>$ 10%), the battery will not be used

to improve the battery life.

4.1.3 Load model

A widely used normal distribution is adopted for describing load fluctuations [195]. Its probability density function is shown in

$$f_{\text{load}}(E_L(n)) = \frac{1}{\sqrt{2\pi}\sigma_L} e^{\frac{1}{2}(\frac{(E_L(n)-\mu_L}{\sigma_L})^2} \ , \tag{4.8}$$

where $E_L(n)$ is the energy consumption from the load during time interval n, μ_L and σ_L are the mean and standard deviation of the load energy.

4.1.4 Scheduling model

The objective of scheduling function is to minimize the total carbon emissions and electricity costs with biases λ_i, that is to minimize J_i in

$$\min_{i,n} J_i = \sum_{n=1}^{N} \sum_{i=1}^{I} j_i(n) \ , \tag{4.9}$$

where

$$j_i(n) = \lambda_i J_{i,\text{cost}}(n) + (1-\lambda_i) J_{i,\text{carbon}}(n) \ , \tag{4.10}$$

$$J_{i,\text{cost}}(n) = \text{Reward}(F_{i,\text{cost}}(n)) \ , \tag{4.11}$$

$$J_{i,\text{carbon}}(n) = \text{Reward}(F_{i,\text{carbon}}(n)) \ , \tag{4.12}$$

$$\lambda_i \in [0,1], \forall i \in \mathbb{N} \ , \tag{4.13}$$

where the objective function is J_i, λ_i is the bias towards carbon emissions and electricity costs, belonging to [0,1], i represents the i th client. $J_{i,\text{cost}}$(n) and $J_{i,\text{carbon}}$(n) are the reward of the cost and carbon respectively. Reward$(\cdot)$ is a function to generate the rewards $J_{i,\text{cost}}(n)$ and $J_{i,\text{carbon}}(n)$ according to $F_{i,\text{cost}}(n)$ and $F_{i,\text{carbon}}(n)$ and historical data in the database, items with lower electricity costs and carbon emissions can get higher rewards, and vice versa. $F_{i,\text{cost}}(n)$ and $F_{i,\text{carbon}}(n)$ are the electricity costs and carbon emissions of the i-th client. The $F_{i,\text{cost}}(n)$ and $F_{i,\text{carbon}}(n)$

are shown in (4.14) and (4.15).

$$F_{i,\text{cost}}(n) = E_{i,\text{bought}}(n)Pr(n) \ , \tag{4.14}$$

$$F_{i,\text{carbon}}(n) = E_{i,\text{bought}}(n)Ca(n) \ , \tag{4.15}$$

where

$$E_{i,\text{bought}}(n) = E_{i,cb}(n) + pe_i(n)E_{i,L}(n) \ , \tag{4.16}$$

$$pe_i(n) = \{1, 0\}, \forall i \in \mathbb{N}, \forall n \in \mathbb{N} \ , \tag{4.17}$$

$$Pr(n), Ca(n), E_{i,cb}(n), E_{i,L}(n) \geq 0, \forall n \in \mathbb{N}^+ \ , \tag{4.18}$$

where $E_{i,\text{bought}}(n)$ is the total energy bought from the main power grid during time interval n. $Pr(n)$ and $Ca(n)$ are the electricity price and carbon emission data during time interval n. Here, $pe = 1$ if the main power grid is used to power the loads, and $pe = 0$ if the batteries are used. The optimization in (4.9) turned to choose the best opportunity (when $Pr(n)$ and $Ca(n)$ are relatively lower) to purchase electricity ($E_{i,\text{bought}}(n)$), thus minimizing $F_{i,\text{cost}}(n)$ and $F_{i,\text{carbon}}(n)$. For this optimization, the constraints of power flow are shown in (4.19) and (4.20).

$$P_{i,\text{net}}(\tau) = P_{i,\text{charge}}(\tau) - P_{i,\text{discharge}}(\tau) \ , \tag{4.19}$$

$$P_{i,\text{net}}(\tau) + P_{i,L}(\tau) = P_{i,\text{PV}}(\tau) + P_{i,cb}(\tau) \ , \tag{4.20}$$

that means the power supply meets the power demand. Where the symbol P means power, τ refers to current time. $P_{i,\text{net}}(\tau)$ is the net power of batteries, $P_{i,\text{charge}}(\tau)$ is the charging power, $P_{i,\text{discharge}}(\tau)$ is the discharging power, $P_{i,L}(\tau)$ is the load power, $P_{i,\text{PV}}(\tau)$ is the PV power and $P_{i,cb}(\tau)$ is the power flow from the main grid to charge the battery.

Besides the photovoltaic panels constraints (4.2), batteries constraints (4.6 and 4.7), charging constraints (4.18), and power balance (4.19 and 4.20), assume no other factors can affect the stability and safety of the microgrid.

4.2 LSTM module

This algorithm aims to predict the states in the DQN using data from recent hours, including the averaged energy generated from the PV, the loads, the carbon emissions, and electricity price, in the next time slot for the DQN. Since LSTM is suitable for time series data prediction, and the aforementioned data all belongs to the time series data, the LSTM architecture is used in conjunction with DQN. The structure is a standard LSTM that is introduced from (2.8) to (2.14).

LSTM algorithms can work with the federated learning algorithm (for example, FedAvg) to form a distributed prediction framework for PV, electricity prices, carbon emissions, and home loads. Instead of uploading the private data, the model parameters including input weights, recurrent weights and fully connected weights of LSTM are uploaded to protect personal data privacy.

4.3 DQN module

As introduced in Chapter 2, there are various possible deep reinforcement learning algorithms that can be used with representative algorithms including DQN, Proximal Policy Optimization, and Deep Deterministic Policy Gradient. Since Proximal Policy Optimization and Deep Deterministic Policy Gradient are both based on the Actor-Critic architecture, they require the establishment of two or even more deep learning models in federated learning, which increases the instability of federated learning. Therefore, for the sake of stability, DQN is chosen as the scheduling algorithm.

The scheduling model proposed in section 4.1.4 can be represented as a Markov Decision Process and solved by a DQN [133].

4.3.1 Markov decision process

Reinforcement learning aims to maximize cumulative rewards, which is consistent with the Markov decision process, so DQN is often modelled with the Markov decision process. Policy is needed in reinforcement learning. Mathematically, the policy

is a mapping, as shown in

$$\pi : State \rightarrow Action \ , \tag{4.21}$$

where π is the policy, a mapping between state and action space, given any state to π, the optimal action can be obtained by mapping π in real-time. In contrast, in traditional solvers like the generic algorithm or Bayesian optimization, re-optimizations are needed whenever the state changes. In a distributed optimization problem like (4.9), the policy can be reused and transferred with similar tasks, so reinforcement learning is chosen, and (4.9) is transferred and described as a Markov decision process. The elements are described as follows:

- Environment: A home microgrid system with loads, PV panels and an energy storage system (batteries).
- State: The state space $\mathbb{S}_i(n)$ can be described in

$$\mathbb{S}_i(n) = [SoC_i(n), Ca(n), Pr(n), E_{i,\mathrm{PV}}(n), E_{i,L}(n), E_{i,\mathrm{MG}}(n)] \ , \tag{4.22}$$

  where the SoC of the battery is $SoC_i(n)$, carbon emission is $Ca(n)$, electricity price is $Pr(n)$, energy from solar irradiance power is $E_{i,\mathrm{PV}}(n)$, load energy consumption is $\mathrm{E}_{i,L}(n)$ and the energy consumption from the main grid is $\mathrm{E}_{i,\mathrm{MG}}(n)$.
- Action: The action space can be described in

$$\mathbb{A}_i(n) = [pe_i(n), E_{i,\mathrm{bought}}(n)] \ , \tag{4.23}$$

  where $E_{i,\mathrm{bought}}(n)$ decides whether to buy electricity into the battery or not, and if so, how much electricity to buy. Also, the $pe_i(n)$ decides to use batteries or electricity from the grid to power the home microgrids.
- Reward: Two scores are set to optimize the accumulated carbon emissions and electricity costs, including environmental and rule scores.
  1. Environmental score: When the carbon emission is lower, and the electricity cost is lower, the more electricity purchased, the higher the reward,

and vice versa.

2. Rule score: Calculate the carbon emission and electricity cost ranking according to historical data. If it is in the high position of low carbon emission and low electricity cost, the more electricity purchased, the higher the reward, and vice versa.

There is a parameter to determine the bias towards these two scores.

To ensure the safety of the microgrid in reinforcement learning, all reinforcement learning actions adhere to the model constraints, such as 4.2, 4.6, 4.7, 4.13, 4.18, 4.19, and 4.20. Please note that the model is based on an ideal microgrid model introduced in this chapter and does not build on strict frameworks like MatPower [196].

The proposed Markov decision process can be solved by deep-Q network in section 4.3.2.

4.3.2 Deep-Q network

The algorithm steps of DQN are described as follows:

- Initialization: This step includes data processing, initialization of the environment, state and action space, and the logic for calculating the energy consumption and PV output every half an hour.
- Observation period: The first N rounds are the observation period. The network is not trained during this period but only to generate data for the 'Memory Replay' in reinforcement learning.
- Making decisions: Selecting half an hour from the dataset randomly. A DNN fits the Q table (The table of each state-action value) to find the optimal action according to the actual situation. The inputs are:
  - Battery energy at the beginning of this half an hour.
  - Carbon emission in this half an hour.
  - Ranking of carbon emission compared with the past 100 hours.
  - Electricity price in this half an hour.

- Ranking of electricity price compared with the past 100 hours.
- Energy generated from the solar irradiance power in this half an hour.
- Load consumption in this half an hour.
- How much energy should buy in this half an hour.
- Whether to use battery or grid energy to supply household usage.

The output of DNN is the Q-table of DQN. Then the DNN will be used to find the best strategy π, that is the mapping from state space $\mathbb{S}_i(n)$ to the action space $\mathbb{A}_i(n)$ in the continuous 100 hours after the selected half an hour. Finally, the trained data will be recorded in the 'Memory Replay' array.

- Training and making predictions: Training the DNN with the data in the 'Memory Replay' array as input and the output of Bellman equations as output. The Bellman equations are given by

$$\mathbb{B} = \alpha \times Q_{\max}(n, \mathbb{S}, \mathbb{A}, W_{\text{DNN}}) + (1-\alpha)(rd + \gamma \times Q_{\max}(n+1, \mathbb{S}, \mathbb{A}, W_{\text{DNN}})) \ , \tag{4.24}$$

where $\mathbb{B}$ is the output of the Bellman equation, W_{DNN} means using the DNN to get the Q-value. rd is the instant reward, $Q_{\max}$ is the maximum Q-value considering all the possible situations. α is the bias towards instant rewards or future rewards and γ is the discount factor of DQN. The training network aims to minimize carbon emissions and electricity costs in 100-hour periods (more than 4 days). Please note that the $Q_{\max}(n+1)$ needs predictions for the environments (including PV generation, loads, electricity price and carbon emission) in the next time slot in the section, so prediction in the section 4.2 is needed.

4.4 Federated learning module

Both load forecasting with LSTM in the section 4.2 and scheduling with DQN in the section 4.3 utilize private household electricity data. If the malicious attackers gain home electricity data, algorithms such as non-intrusive load monitoring can

be used to infer the daily activity patterns of users, compromising their privacy [197]. One method to protect privacy is to use federated learning. Both LSTM and DQN can utilize federated learning to protect private data [75]. In the case of LSTM, parameters can be uploaded to the server. For DQN, a DNN is employed to approximate the Q-table within the DQN. The parameters of this DNN can be uploaded to the server for federated learning.

4.4.1 Federated learning-based LSTM

1. Initialization: The server-side initializes the global LSTM network structure and parameters, then sends that to each client, including all the weights and biases of the input, output and forgotten gates.

$$w \leftarrow W_{\text{in}}, W_{fg}, W_{gc}, W_o, W_v, b_{\text{in}}, b_{fg}, b_{gc}, b_o, b_U \ . \tag{4.25}$$

2. Local training: The client i conducts n-rounds local training according to the local data

$$w_i(n+1) \leftarrow w(n) - \eta_{lr} \nabla L(w(n)) \ , \tag{4.26}$$

where η_{lr} is the learning rate and $\eta_{lr} \nabla L(w(n))$ is the batch gradient, the superscript i represents the i-th client. n indicates the time slot.

3. Uploading weights: The client sends all the trained weights, biases and other parameters $w(n+1)$ of the LSTM model to the server.

4. Averaging: The server checks whether the data of all the clients are received. If some clients fail, they are required to retransmit. If all weights of clients are received, the server calculates the average weight of each client and then sends the averaged model to each client. The averaging process is described by

$$w(n+1) \leftarrow \sum_{i=1}^{I} \frac{DtV_i}{DtV} w_i(n+1) \ , \tag{4.27}$$

where DtV_i and DtV refer to the data volume of each client and total data volume.

5. Training: Repeating steps 2-4 until reach convergence.

4.4.2 Federated learning-based distributed DQN algorithm

Two improvements were made to the original DQN algorithm: firstly, the DQN algorithm is integrated with federated learning. In traditional DQN, a DNN is typically used to approximate the Q-table. Through federated learning, the parameters of this DNN can be uploaded to the server for collaborative training. Secondly, multi-objective optimization with a Pareto front was introduced in DQN. As both carbon emissions and cumulative electricity costs need to be optimized simultaneously, when a new Pareto front agent emerges, the models of agents can be stored and updated.

The pseudocode of the proposed algorithm is described in Algorithm 1, where lines 1-6 are the initialization, including the loading of solar irradiance, electricity price, and carbon emission data from datasets, and the initialization of states, load profiles, reply memories $\mathbb{M}_r$ on capacity $\mathbb{C}_1$, and Pareto memories $\mathbb{M}_p$ on capacity $\mathbb{C}_2$, and weights w_{target} and w_{online} for the target network $\text{DQN}_{\text{target}}$ as well as online network $\text{DQN}_{\text{online}}$ of DNNs, the introduction of target network makes the network more stable. The proposed network will randomly select a client as the server. Line 7 set up a outer loop for different training episodes. Line 8 is the initialization of the environment and states for the deep-Q network. Line 9 set up a inner loop for training the proposed DQN in each training episode. Line 10 is the epsilon-greedy algorithm. With a probability of ϵ, choose a random action, and with a probability of $1 - \epsilon$, choose the action with the highest Q-value according to DQN. Line 11 and Line 12 are a step forward for the deep-Q network algorithm. The online neuro network decides the action. Both of them are used during the observation period of the DQN to collect some data for initialising memory replies and they are also used during the real training periods. Line 13 stores the experience for future training. Line 14 calculates the accumulated carbon emissions and electricity bills. Lines 16-18 judge whether this online network is a non-dominated solution according to carbon emissions and electricity costs. If so, store the results. 19-21 is to update the online network with the target network regularly (every Tr_{gap} rounds).

22-26 train the target network with the stored memory regularly (every Tr_{renew} rounds), with line 24 using Bellman's equation to estimate the possible best Q-value $\mathbb{B}_{i,mb}$ and line 25 perform gradient descent. The server performs lines 31-37 to perform federated learning regularly (every Tr_{upload} rounds), including aggregation and generating overall Pareto fronts, and otherwise, the clients perform lines 28-30 to upload the weights of deep neuro network and Pareto fronts to the server, the server performs aggregation operations. In line 40, the output is a list containing all the DQN models and its parameters in the Pareto front.

4.4.3 Algorithm time complexity

Excluding considerations for training rounds and time slots, the proposed algorithm has the following complexities for each computational step.

1) DNN for approximating Q-table: The algorithm time complexity is $\mathbb{O}(d_{\text{in}} \times d_{\text{out}})$ for each layer, where d_{in} and d_{out} are the input and output dimensions [185].

2) Epsilon-greedy algorithm: The epsilon-greedy algorithm has the probability ϵ for random action selection and 1-ϵ for selecting the action with the maximum expected reward. For random action selection, the algorithm time complexity is $\mathbb{O}(1)$; for selecting the action with the maximum expected reward, the forward pass algorithm time complexity is $\mathbb{O}(d_{\text{in}} \times d_{\text{out}})$, assuming the combination of different action space sizes is $\mathbb{S}_{\mathbb{A}}$, resulting in a total time complexity of $\mathbb{O}(\mathbb{S}_{\mathbb{A}} \times d_{\text{in}} \times d_{\text{out}})$ [137].

3) Experience replay for random data retrieval: The algorithm time complexity is $\mathbb{O}(1)$ for data retrieval; storage complexity depends on the number of updates, assuming K data updates, with a constant time complexity of $\mathbb{O}(K)$ [137].

4) Target Network Update: The algorithm time complexity is $\mathbb{O}(N_{\text{PC}})$ for copying parameters from the online network to the target network if the total parameter volume of the weights and bias are N_{PC} [137].

In the proposed algorithm, there are two improvements:

5) Finding Pareto fronts: Firstly, sorting all the data needs a minimum $\mathbb{O}(N_{\text{PF}} \times \log N_{\text{PF}})$, where N_{PF} is the length of the Pareto front data. Then filtering the Pareto front needs an extra $\mathbb{O}(N_{\text{PF}})$ to judge the top items after sorting and getting the final fronts.

Algorithm 2: Proposed federated multi-objective deep Q-learning algorithm with Pareto fronts

Load data from datasets
Initialize the state: Client i_1 - i_3 or Server
Initialize home microgrid loads with (4.8), with Tr_{gap}, Tr_{renew} and Tr_{upload}
Initialize replay memory $\mathbb{M}_r$ to $\mathbb{C}_1$ and pareto memory $\mathbb{M}_p$ to capacity $\mathbb{C}_2$
Initialize target network $\text{DQN}_{\text{target}}$ with random weights w_{target}
Initialize online network $\text{DQN}_{\text{online}}$ with random weights w_{online}
for *episode* $= 1, M$ **do**
    Initialise sequence $s_i(1) = \{data_i(1)\}$ with datasets and preprocessed sequenced $\phi_i(1) = \phi\left(s_i(1)\right)$
    for $n = 1, N_{DQN}$ **do**
        With probability ϵ select a random action $act_i(n)$ otherwise try all actions act and select $act_i(n) = \max_{act} \text{DQN}_{\text{online}}\left(\phi\left(s(n)\right), act; w_{i,\text{online}}(n)\right)$
        Execute action $act_i(n)$ and observe reward $j_i(n)$ from (4.9) then get $data_i(n+1)$
        Set $s_i(n+1) = s_i(n), act_i(n), data_i(n+1)$ and preprocess $\phi_i(n+1) = \phi\left(s_i(n+1)\right)$
        Store $(\phi_i(n), act_i(n), j_i(n), \phi_i(n+1))$ in $\mathbb{M}_r$
        Calculate accumulated carbon emissions $CE_{i,\text{total}}$ and electircity costs $EC_{i,\text{total}}$ with $F_{i,\text{cost}}(n)$ and $F_{i,\text{carbon}}(n)$
    end for
    if *($CE_{i,total}$ and $EC_{i,total}$ non-dominated)* **then**
        Store and update Pareto fronts with w_{target} in $\mathbb{M}_p$
    end if
    if *(!episode%Tr_{gap})* **then**
        $\text{DQN}_{\text{online}} = \text{DQN}_{\text{target}}$
    end if
    if *(!episode%Tr_{renew})* **then**
        Sample random minibatch of transitions $(\phi_{i,mb}, act_{i,mb}, j_{i,mb}, \phi_{i,mb+1})$ from $\mathbb{M}_r$
        Set $\mathbb{B}_{i,mb} = j_{i,mb} + \gamma \max_{act'} \text{DQN}_{\text{target}}\left(\phi_{i,mb+1}, act'; w_{\text{target}}\right)$
        Perform a gradient descent step on $\left(\mathbb{B}_{i,mb} - \text{DQN}_{\text{target}}\left(\phi_{i,mb}, act_{i,mb}; w_{i,\text{traget}}\right)\right)^2$ for $\text{DQN}_{\text{target}}$.
    end if
    if *(!episode%Tr_{upload})* **then**
        if *(State==Client)* **then**
            Upload w_{target} and $\mathbb{M}_p$ to Server
        end if
        if *(State==Server)* **then**
            Collect weights $w_{i,\text{target}}(n+1)$ from Clients
            Calculate the data volume DtV_i of Client i
            Calculate the total data volumn DtV
            $w_{\text{target}}(n+1) \leftarrow \sum_{i=1}^{I} \frac{DtV_i}{DtV} w_{i,\text{target}}(n+1)$
            Generate Pareto fronts with $\mathbb{M}_p$, Return global model $w_{\text{target}}(n+1)$
        end if
    end if
end for
Output: The $\text{DQN}_{\text{online}}$ of Pareto fronts

6) Federated learning part: The algorithm time complexity for model aggregation at the server is $\mathrm{O}(N_{\mathrm{PC}} \times N_c)$, where N_{PC} is the number of parameters and N_c is the number of clients [75].

In summary, although the proposed algorithm introduces additional steps such as Pareto front filtering $\mathbb{O}(N_{\mathrm{PF}} \times \log N_{\mathrm{PF}}) + \mathbb{O}(N_{\mathrm{PF}})$ and parameter copying $\mathbb{O}(N_{\mathrm{PC}} \times N_c)$, the overall algorithm time complexity is still in the same order of magnitude as the standard DQN, which is $\mathbb{O}(d_{\mathrm{in}} \times d_{\mathrm{out}}) + \mathbb{O}(S_{\mathbb{A}} \times d_{\mathrm{in}} \times d_{\mathrm{out}}) + \mathbb{O}(K) + \mathbb{O}(N_{\mathrm{PC}})$, especially in scenarios with a limited number of clients. Therefore, it does not impose a significantly higher computational burden compared to the original algorithm.

4.5 Simulation setup

This section describes the simulation environment, including the parameters of the microgrid, prediction system, and the DQN scheduling algorithm.

For software, the proposed microgrid, LSTM, DQN and federated learning were developed in MATLAB with toolboxes. For hardware, 16GB NVIDIA GeForce RTX 3080 GPU was used with Intel i7-12700H CPU and 32GB memory.

4.5.1 Microgrid parameters

The simulation parameters of the microgrid are shown in table 4.1. For testing the algorithm, it is assumed that there is no conversion loss.

Table 4.1: Microgrid simulation parameters

Parameters	**Values**
Default time interval	0.5 hour
PV panel area	$1m^2$
Conversion efficiency	20%
Capacity of the battery	4×50Ah
Rated voltage	12V
Conversion or storage loss	0%
Maximum energy bought every half an hour	each battery 5Ah

4.5.2 Load forecasting parameters

For the load profile, the typical UK household electricity demand (around 10 Kw a day, with a 15% normal distribution standard deviation fluctuation) is used for the simulations, and this load profile is in accordance with the distribution in [198]. The distribution of the electricity prices is shown in Figure 4.3. The peak load occurs at 8 AM and 8 PM, respectively.

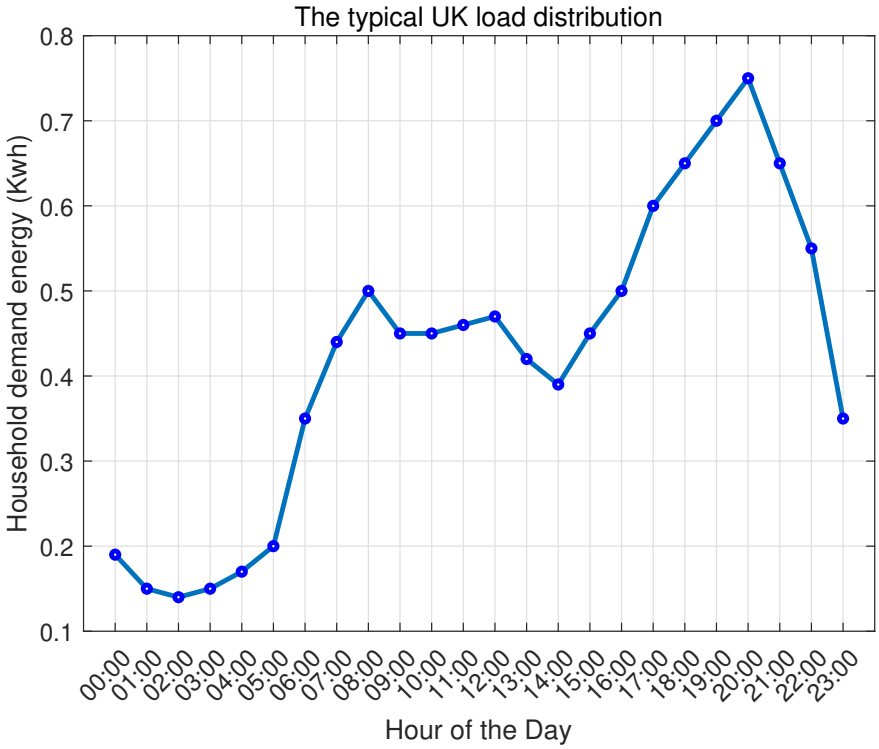

Figure 4.3: The typical load distribution of the UK families.

The load predictions will be introduced in 4.6.1, its parameter selections are shown in table 4.2.

4.5.3 PV prediction parameters

The PV datasets (2017 to 2020) from Durham, England, UK were used, with a manual setting of a positive or negative 5% random PV fluctuation [199,200]. After computing the energy obtained from the PV panels via the proposed model, the energy data from each household is utilized for federated learning.

The tuned LSTM parameters and federated learning parameters are recorded

Table 4.2: LSTM and federated learning simulation parameters for load predictions

Parameter in LSTM and federated learning	**Values**
LSTM hidden nodes	50
LSTM learning rates	0.01
LSTM training rounds	800
LSTM optimizer	Adam
LSTM loss function	Mean-Squared-Error
LSTM state activation function	tanh
LSTM gate activation function	sigmoid
Federated learning client number	4
Federated learning local training rounds	10

in table 4.3. The model's predictions as well as the parameter selections will be introduced in 4.6.2.

Table 4.3: LSTM and federated learning simulation parameters for PV forecasting

Parameter in LSTM and federated learning	**Values**
LSTM hidden nodes	200
LSTM learning rates	0.001
LSTM training rounds	800
LSTM optimizer	Adam
LSTM loss function	Mean-Squared-Error
LSTM state activation function	tanh
LSTM gate activation function	sigmoid
Federated learning client number	4
Federated learning local training rounds	10

4.5.4 Carbon emission and electricity price datasets

The north-east England wholesale electricity price datasets (2020) from Energy Stats UK were used in this thesis. The wholesale electricity prices are assumed to be equal to household electricity prices. Therefore, in the following simulations, the wholesale electricity data is used for the households, replacing the household electricity data. The UK general carbon emissions datasets (2020, including predicted data) from Carbon Intensity were used [201, 202].

The distribution of the price data is shown in 4.4. It can be observed that there is a peak in electricity prices between 3 PM and 6 PM.

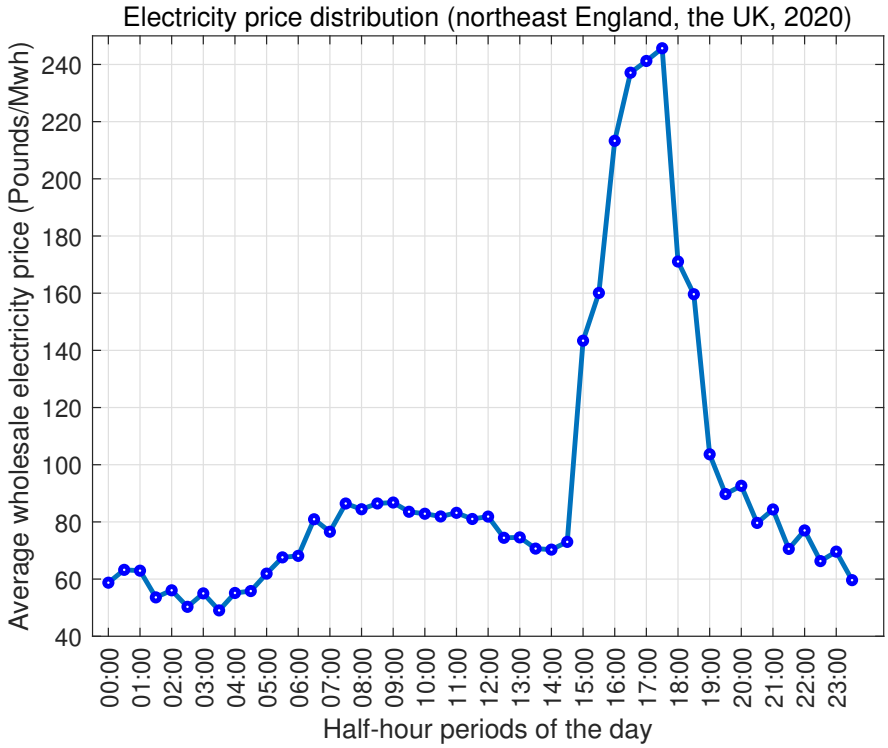

Figure 4.4: The wholesale electricity price distribution of the north-east UK.

The distribution of the carbon emissions data is shown in Figure 4.5.

4.5.5 DQN parameters

The super-parameter of the DQN is summarized in table 4.4. The DQN scheduling system depends on both the real and forecasted data, including load, PV, carbon emissions and electricity prices and no extra dataset is needed.

4.6 Results

This section presents the simulation results, including the outcomes of the load and PV predictions and the federated DQN's training results. Further, it explores the motivations behind DQN's scheduling decisions. Finally, simulations are conducted between the proposed federated DQN, the individual DQN, and the comparative

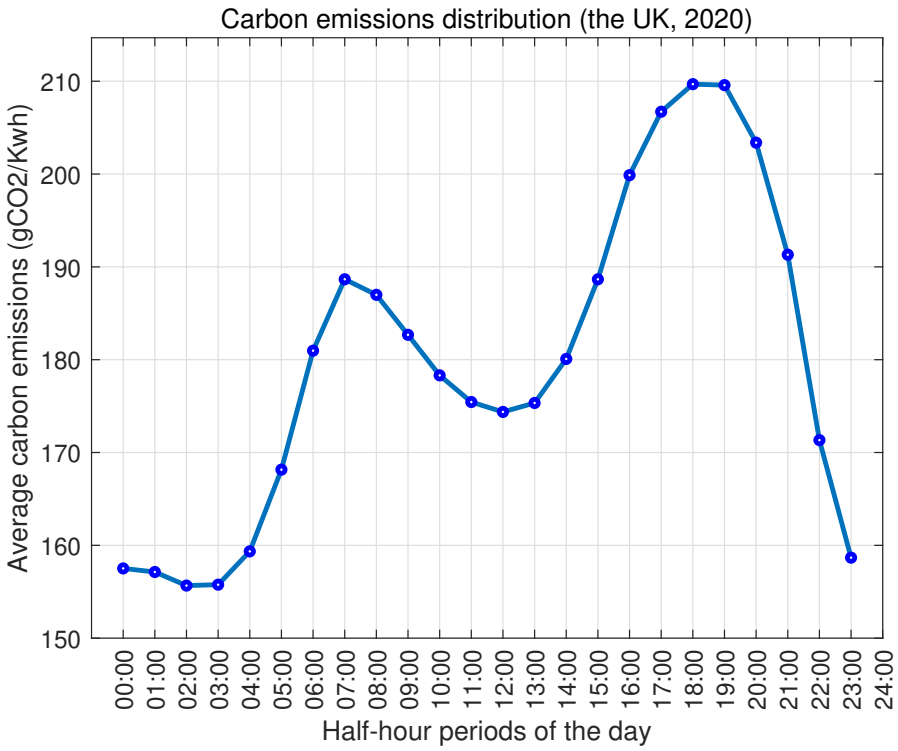

Figure 4.5: The general carbon distribution of the UK.

simulations including the time-based scheduling algorithm as well as the multi-agent DQN. Moreover, the optimization bias towards carbon emissions and cumulative electricity costs reduction can be adjusted by altering the agents in the Pareto front.

4.6.1 Results from load forecasting

In load forecasting, a federated learning prediction system consisting of four home clients is considered. In the simulations, Client 1 is selected as the Server to perform aggregation, receiving and calculating the average of all trainable parameters in the federated learning. After each round of training, Client 1 broadcasts its model to the other Clients. In the following comparative simulations, all the results shown in the simulations are only from Client 1 in their federated learning. Therefore, the results of the other clients are not shown because they have a similar prediction performance. All clients share the same architecture, optimizer, and loss function. The choice of the optimizer (Adam), and the loss function (Mean Squared Error) is because these are commonly used for regression problems.

Table 4.4: DQN Simulation parameters

Parameters in DQN	**Values**
Training iterations	20000
DNN input Nodes	7
DNN output Nodes	1
DNN hidden Nodes	80× 80
Training function	Levenberg-Marquardt
Reply memory size	400000
Initial exploration	1
Final exploration	0.05
Update frequency	400 training steps
Observation period	200 training steps
Mini-batch size	200 rounds, 100 hours simulation

To determine the optimal parameters for load forecasting, the network structure was selected as the first target for parameter tuning, and its size was adjusted. The hidden layer architecture of the LSTM was simulated by starting from 50 neurons and increasing in increments of 50, up to a maximum of 300 neurons. Figure 4.6 shows the results for 50, 100, 150, and 200 neurons, no convergence results were obtained for 250 and 300 neurons. It was found that 50 or 100 neurons were the best choices for the simulations. The simulations also found that changing the network size did not significantly affect the prediction performance, with all of them having nearly consistent results between the 50 and 200 neurons.

To determine the optimal learning rate, it was necessary to adjust and simulate this learning rate. With the LSTM hidden layer set to 50 neurons, the simulations utilized learning rates of 0.1, 0.01, 0.001, and 0.0001. As shown in Figure 4.7, the simulation found that 0.01 yielded the best results. The simulation discovered that adjusting the learning rate could affect the prediction results to only a small extent. Figure 4.8 shows the learning process of the federated LSTM with 50 hidden layer neurons and a learning rate of 0.01, while displaying the prediction results for the four clients.

Subsequently, the federated LSTM was compared with the federated DNN and federated CNN architectures. For the DNN, the same hidden neurons and learning rate were used. For the CNN, the hidden layer was fixed at 50 neurons, and the

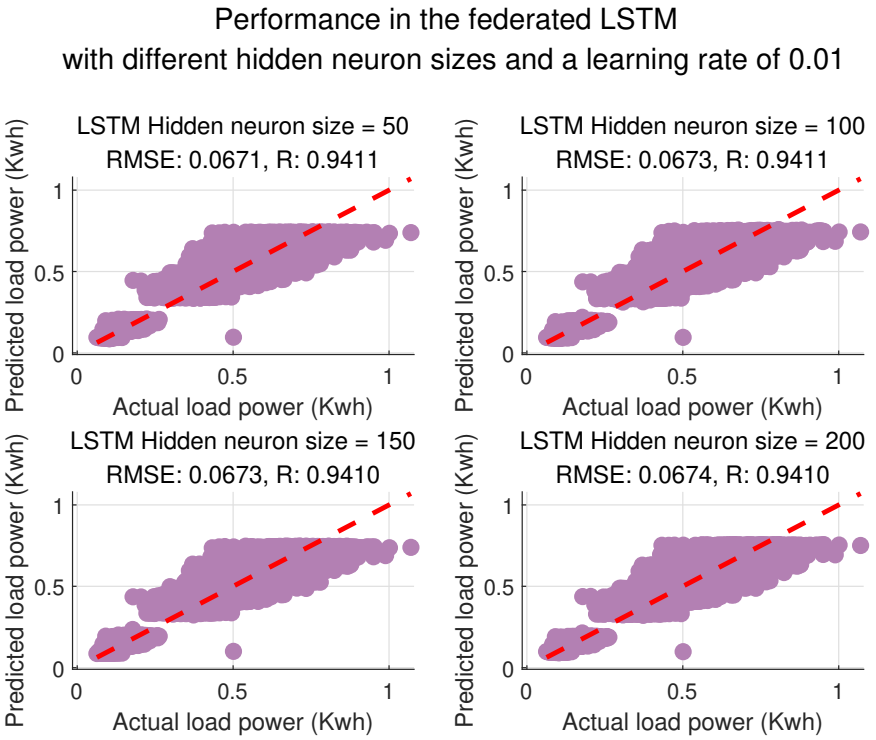

Figure 4.6: The different results for load predictions, produced by changing the LSTM network's hidden neuron size when the learning rate is 0.01.

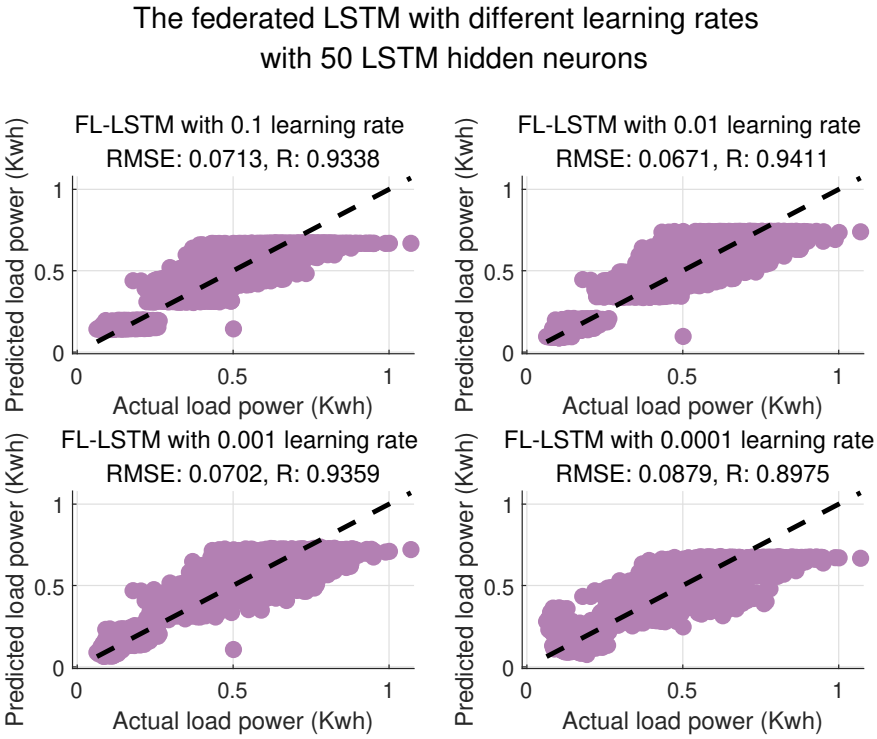

Figure 4.7: Fixing the LSTM hidden neuron size at 50, the load prediction results were obtained by trying different learning rates.

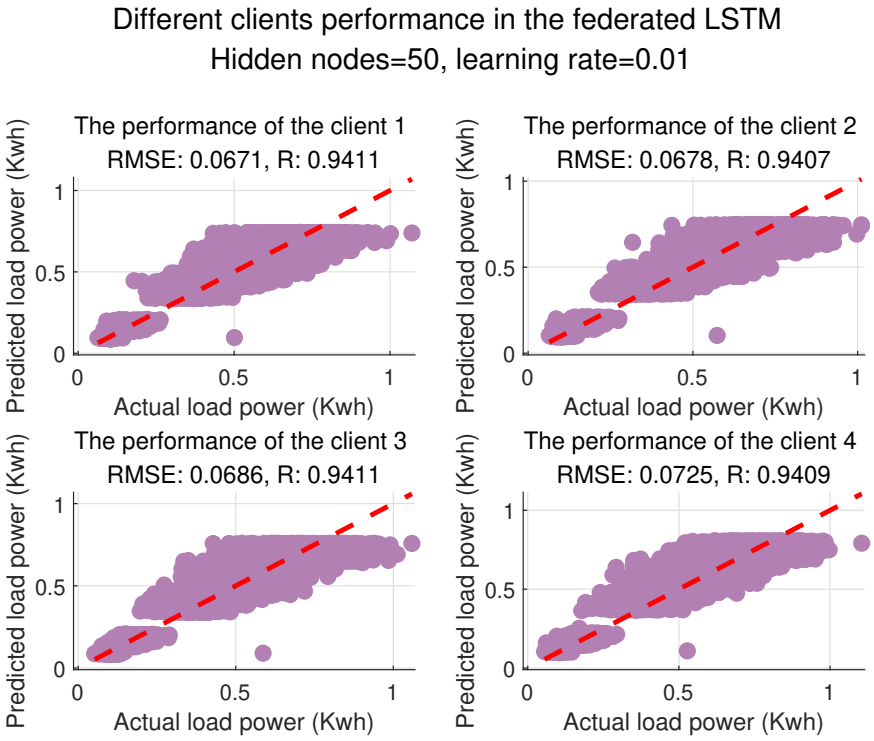

Figure 4.8: The load prediction performance of each client in federated learning when the LSTM hidden size is set to 50 and the learning rate is 0.01.

input size was adjusted to 1×5, with a convolution kernel size of 1×3, so that after the convolutional calculation, the size of the vector (1×3) was the same as the input size of the LSTM and DNN. In Figure 4.9, the simulations found that the results from the CNN and DNN architectures were far inferior to those of the LSTM in terms of prediction performance. Therefore, LSTM is the most suitable architecture for forecasting in the comparative simulations.

4.6.2 Results from PV prediction

Although PV data is publicly available, once solar irradiance is converted into household energy, it can be understood as energy data, hence there is a privacy concern. Therefore, federated learning has been applied to the energy generated by PV from different households.

Photovoltaic forecasting uses the same federated learning architecture as load forecasting. Similarly, each client sends their forecasting model parameters to Client 1, where Client 1 executes the federated learning algorithm and broadcasts the

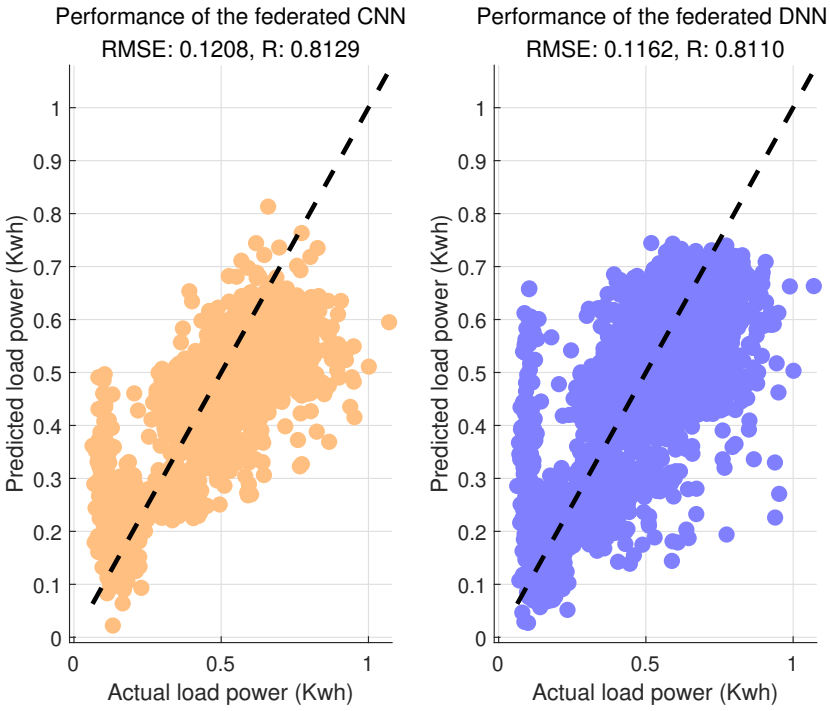

Figure 4.9: In the comparative simulation, the load prediction results are shown in the graph from the DNN and CNN when the hidden size is set to 50 and the learning rate is 0.01.

aggregated model to all the other clients. The comparative simulations only display the predicted results of Client 1.

To find the optimal predictive parameters for the LSTM, the network structure of the LSTM was adjusted first in Figure 4.10. The LSTM architecture utilized 50, 100, 200, and 300 neurons, with 200 yielding the best results. The simulation discovered that the prediction results were not sensitive to the size of the architecture.

To find the optimal learning rate for the LSTM, various learning rates such as 0.1, 0.01, 0.001, and 0.0001 were tested, with the LSTM architecture set to 200 neurons. The simulations found that the prediction was most effective when the learning rate was 0.001, as shown in Figure 4.11. The simulation showed that the learning rate could significantly impact the prediction performance.

With the hidden layer size set to 200 neurons and the learning rate at 0.001, the federated learning prediction results for each client are shown in Figure 4.12.

The federated learning predictions also involve comparisons with the DNN and CNN networks, both using a hidden layer size of 200 and a learning rate of 0.001.

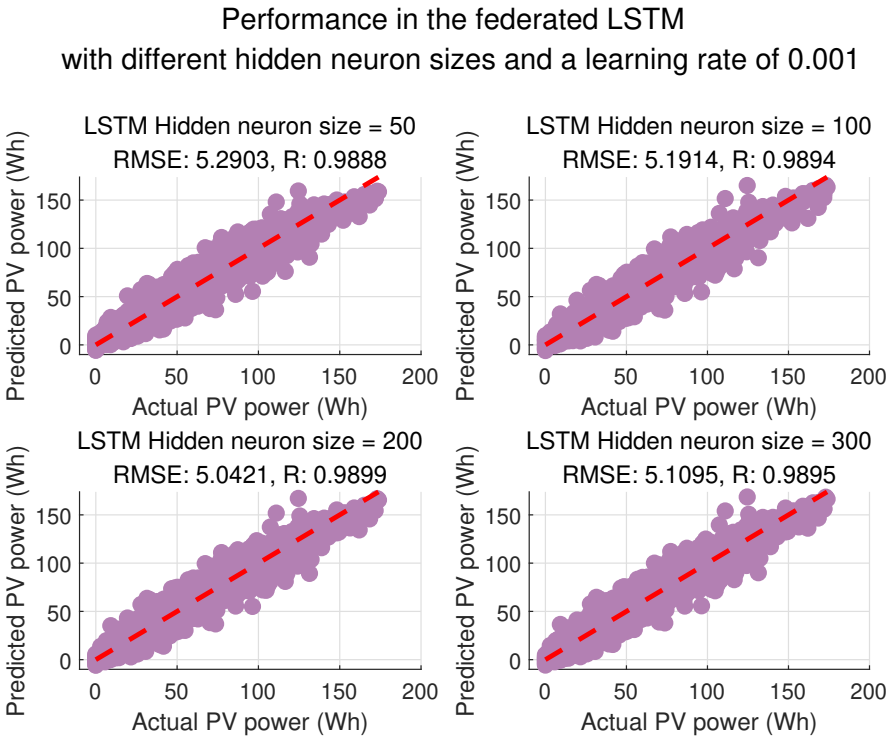

Figure 4.10: The different results for PV forecasting, produced by changing the LSTM network's hidden neuron size when the learning rate is 0.001.

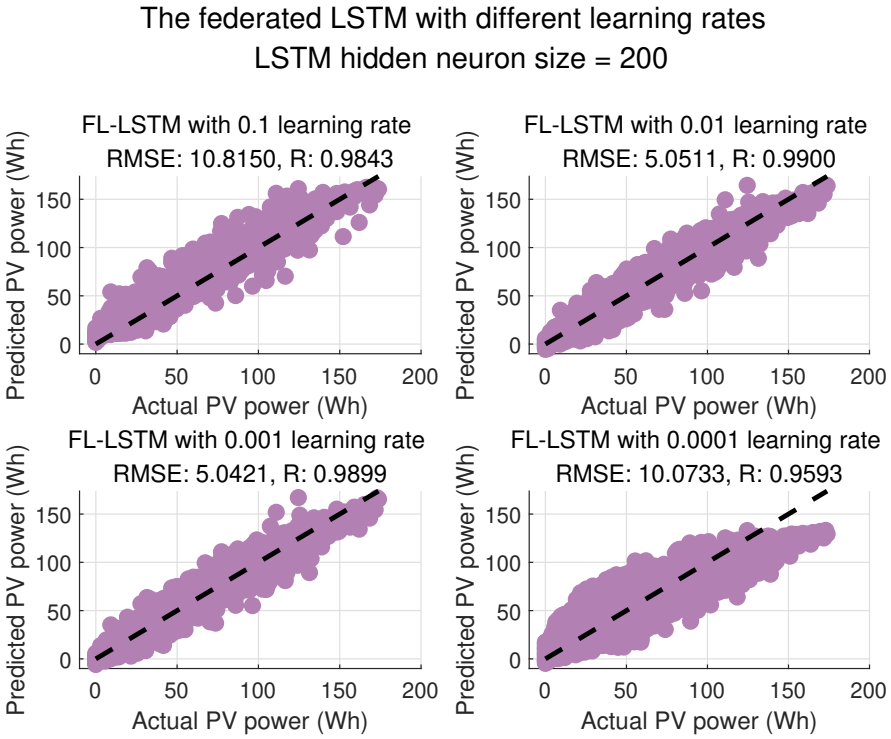

Figure 4.11: Fixing the LSTM hidden neuron size at 200, the PV prediction results were obtained by trying different learning rates.

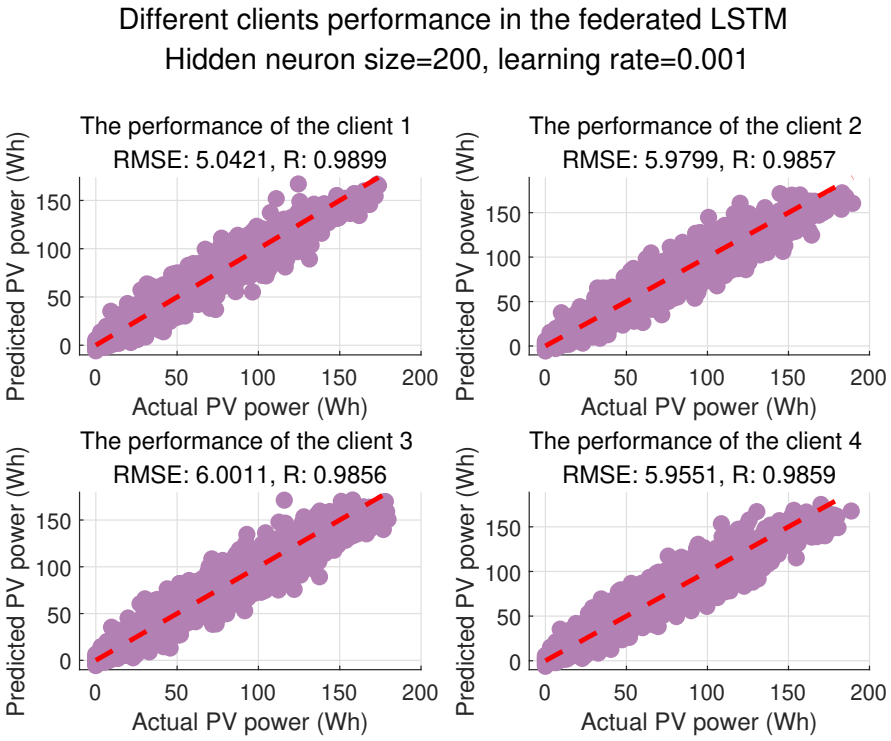

Figure 4.12: The PV prediction performance of each client in federated learning when the LSTM hidden size is set to 200 and the learning rate is 0.001.

For the DNN, the same method as the LSTM was used, predicting the next time slot based on the previous three time slots, meaning the input size was 3×1, and the output was 1×1. For the CNN, since convolutional operations are required, an input size of 3×1 with a kernel size of 1×1 are a bad choice, as it cannot reflect the convolution operation well. For comparison, an input size of 5×1 and a kernel size of 3×1 were used, yielding a result of 3×1, consistent with the input size of both the LSTM and DNN. The results showed that the predictive performance of the DNN and CNN networks was inferior to that of the LSTM, leading to the choice of the LSTM as the best federated predictive network.

4.6.3 Results from carbon emissions and electricity prices prediction

Regarding carbon emissions and electricity prices, since this data is publicly available without any privacy concerns, there is no need to use federated learning for privacy protection. However, normal non-federated predictions are needed for producing the

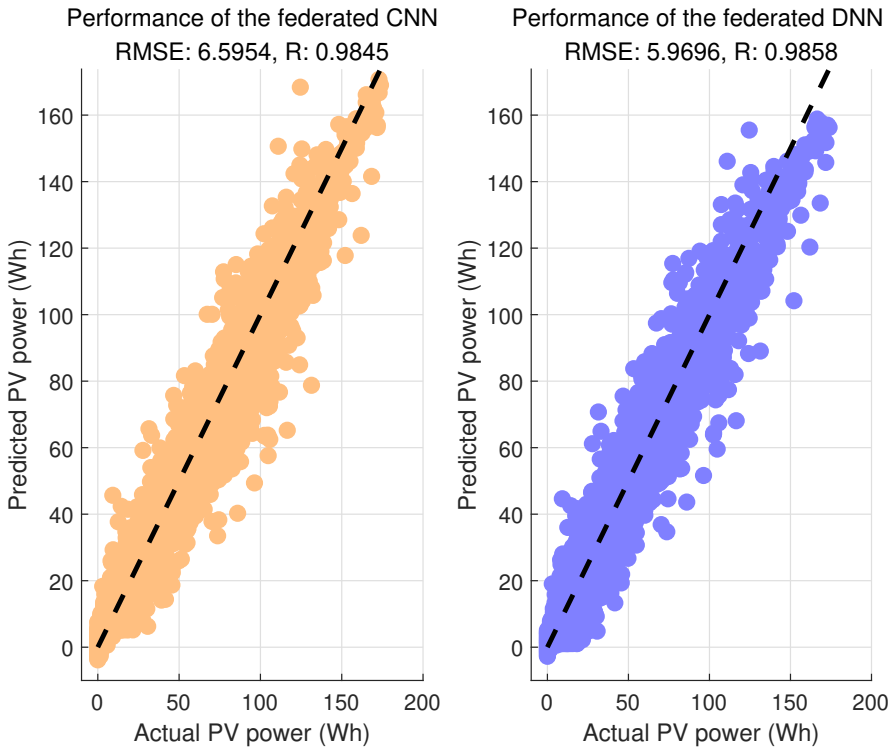

Figure 4.13: In the comparative simulation, the PV prediction results of DNN and CNN when the hidden size is 200 and the learning rate is 0.001.

future environment of the DQN.

For carbon emissions predictions, the prediction value is included in the dataset, which is used for the DQN.

The accurate prediction of electricity prices will impact the performance of the DQN system. In this simulation, the real electricity price data for the next half an hour was used instead of the predicted data. This is because the electricity prices have been changing significantly due to the COVID-19 pandemic and the Ukraine war in recent years, thus making it hard to make accurate predictions [203] [204].

4.6.4 Results from federated DQN convergence

As shown in Fig. 4.14, the graph illustrates the optimization progress of the DQN algorithm over 20,000 training steps, divided into 100 rounds, with 200 training steps in each round. Each point is the average result from over 100 hours, and it is assumed that in the 100th hour, all the remaining energy of the battery is also calculated for both the average electricity price and the carbon emissions. These

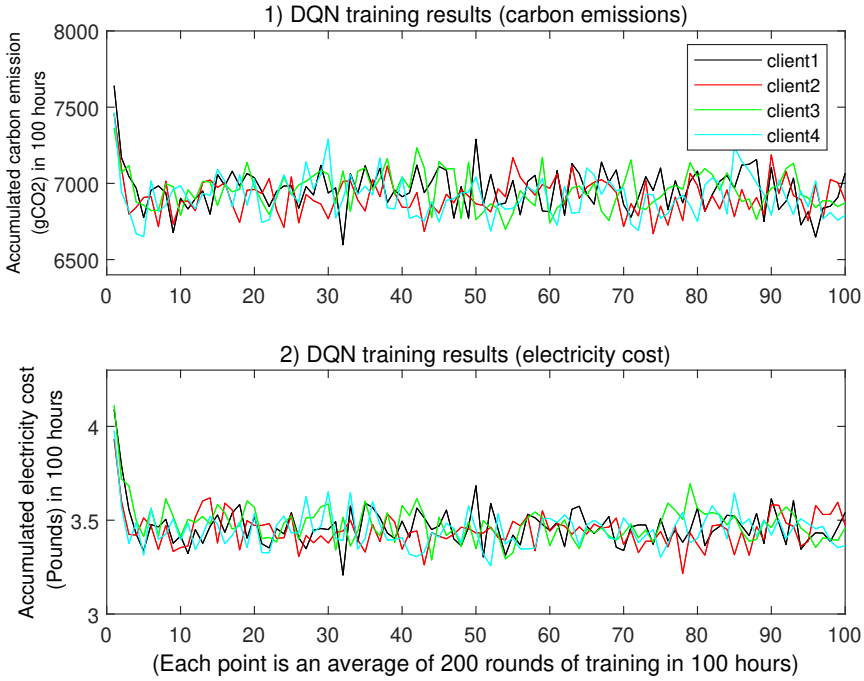

Figure 4.14: Optimizations vs training iterations of a federated DQN, including: 1) The cumulative carbon emissions per 100 hours. 2) The accumulated electricity costs per 100 hours in relation to the number of training rounds. It can be observed that the carbon emissions and electricity costs decrease with the optimization.

calculated values are then deducted from the accumulated value. The upper subplot represents the carbon emissions, showing rapid CO2 decreases at the beginning, with later fluctuations in the CO2 in the following training rounds. The lower subplot displays the total cost of electricity consumption, exhibiting fluctuations but also indicating an overall converging trend.

Through training, it was found that compared to the first round at the beginning, where the average carbon emissions (7480 gCO2) and cumulative electricity costs (4.03 Pounds), the achieved minimum values can reduce cumulative carbon emissions by 11.8% (6597 gCO2) and cumulative electricity costs by 20.6% (3.2 Pounds).

If the impact of the first 200 rounds of observation periods is not included, in comparison with the 200-400 iterations (average carbon emissions was 7070 gCO2, and the average electricity costs were 3.67 Pounds), the achieved minimum values can reduce cumulative carbon emissions by 6.7% (6597 gCO2) and cumulative electricity costs by 12.8% (3.2 Pounds).

The training process shows that the proposed DQN gradually explores and cap-

tures the optimal electricity purchasing opportunities, achieving a bi-objective optimization balance for both carbon emissions and electricity costs.

4.6.5 Results from the scheduling tests

In this part, 100 hours were chosen to evaluate the performance of the trained DQN. There are three subgraphs in Fig. 4.15. The Y-axis labels show that the first and second subgraphs are the carbon emission coefficient and the electricity price every half an hour. The third subgraph is the decision made by the DQN, that is, when and how much electricity to buy.

It is observed that in the first subplot, the hours with lower carbon emissions are around the 25th hour, 47th hour, and 95th to 100th hours. In the second subplot, the times with lower electricity prices are around the 0-th hour, 50th hour, 68th hour, and 95th to 100th hour. It can be observed that the trends of carbon emissions and electricity prices are consistent, which also proves the consistency of using electricity price and carbon emission datasets. In the third subplot, the DQN only bought energy when the carbon emission or electricity price was low, such as in 0th to 2nd, around 10th, 20th to 27th, 45th to 50th, around 70th and 95th to 100th. These time slots are in accordance with the time slots having lower carbon emissions and electricity costs. Therefore, the energy efficiency is improved by the DQN.

It can be found that the proposed DQN can seize the opportunity to buy electricity when carbon emissions or electricity prices are lower than at other times. Moreover, the lower the electricity prices or carbon emissions, the more electricity is purchased, as seen during the 67th and 99th hours.

4.6.6 Case studies and comparisons

Two comparison algorithms and two case studies are considered in this part, for testing the effect of the proposed algorithms.

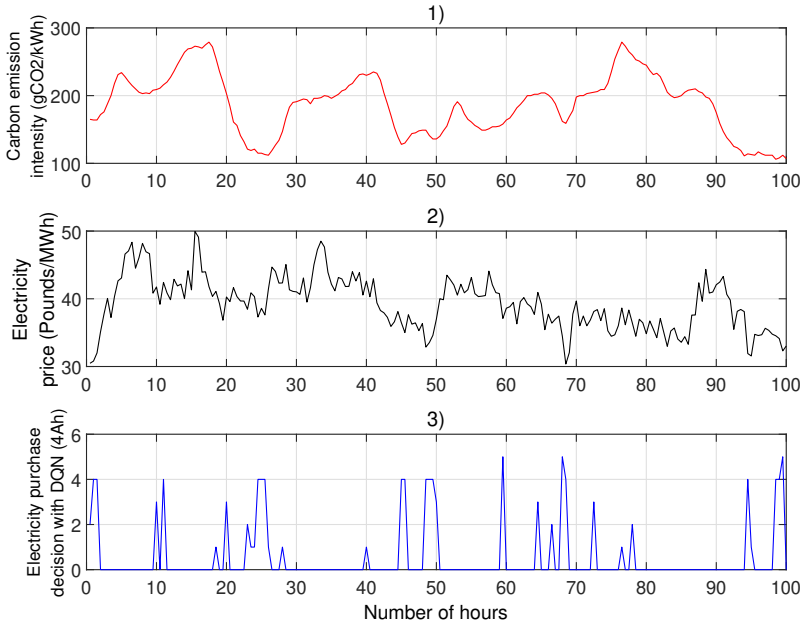

Figure 4.15: A single DQN is used to acquire the best electricity purchase opportunities, including: 1) Carbon emissions for 100-hour intervals. 2) Electricity prices for 100-hour intervals. 3) Electricity purchasing decisions made by the DQN specify when and how much electricity (in Ah) to buy. Simulations have found that the DQN purchases electricity when electricity prices or carbon emissions are low.

Comparisons

This study sets up two comparative simulations: 1. Scheduling based on time, 2. Scheduling based on the multi-agent algorithm.

1) The first scenario is a comparative simulation based on time scheduling, replicating the algorithm from paper [205]. The paper mentioned that by utilizing time-of-use electricity pricing, batteries can be charged during off-peak periods and discharged during peak periods. This approach was adapted into a comparative simulation in this Chapter. According to the distributions in Figures 4.4 and 4.5, the periods when electricity prices and carbon emissions are relatively low are between 10 PM and 5 AM. Since carbon emissions and electricity prices are lower during the nighttime, as much as possible electricity can be purchased during that time, to reduce both carbon emissions and costs. The simulation adopts four time-models for electricity purchasing: 12 AM to 4 AM, 11 PM to 5 AM, 11 PM to 4 AM, and 10 PM to 5 AM. No electricity is purchased during the remaining time periods.

2) The second scenario is a multi-agent model with shared memory data in an authorized third-party database. This plan is an adaptation of the algorithm from paper [206], which employs a framework called multi-agent architecture with centralized training and distributed application in DQN agents. It doesn't simply share real-time data but rather shares data from memory replay, which means the clients share their memory experience with the server. The server trains a global model through global data memory replay and then updates the global model for each client. The clients generate memory experience based on the local data and provide that to the server for future training. The solution poses the risk of privacy leakage due to the use of a third-party database.

Proposed algorithms

The proposed algorithm also has two case studies: 1. Single DQN, 2. Federated DQN. They are described as follow:

Figure 4.16: Single DQN scenario. In the diagram, five homes each conduct their training independently, without communicating with each other. The data or model gradients are not shared between them.

1) The third scenario is a single DQN without sharing data or parameters with each other, as shown in Fig. 4.16. This scenario has the highest privacy protection. However, a lack of data has led to data isolation, preventing the algorithm from benefiting from improvements in the future through the utilization of data from other homes.

2) The forth scenario is shown in Fig. 4.17 (**Algorithm 2**), the DQN with federated learning. The designed DQN is decentralized, meaning any home/client

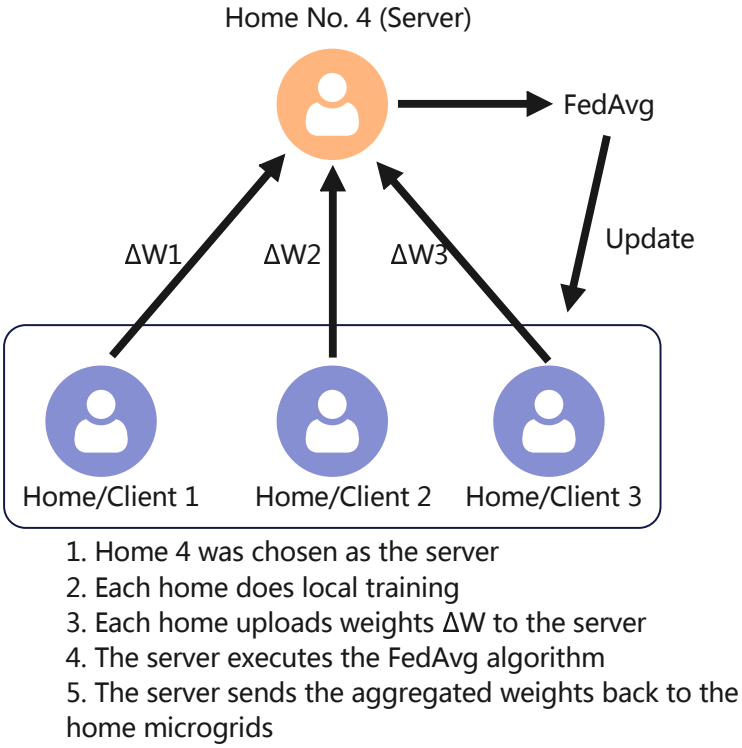

Figure 4.17: Federated learning scenario. The home (client 4) is selected to act as the server in the diagram. Other homes (clients) transmit their model parameters to it for federated learning.

can become the server. For instance, in Fig. 4.17, Home 4 is the server, while Homes 1-3 act as clients. Home 4 receives model parameters from all clients for federated learning in this setup. The advantage of this approach is to mitigate the impact of server failures. Instead of the data, the federated learning shares the weights and biases of the DQN, and privacy is protected because there is no need to upload private data.

Apart from these algorithms, all comparative simulations adhere to the proposed DQN model and its constraints, namely the photovoltaic model in section 4.1.1, the battery model in section 4.1.2, the load model in section 4.1.3, and the power constraints in formulas 4.19 and 4.20.

Simulations are conducted based on the four scenarios above. Scenario one gets the results from the rules, so there is no Pareto front from the optimizations, and the averaged carbon emissions, as well as electricity costs, should be used for making comparisons instead of the optimal values. The results are compared with the Pareto front values in the second, third and fourth scenarios because the results are directly

from the DQN optimizations. There are two objectives: The optimization of both carbon emissions and the costs of electricity. During the simulation, the network parameters of the agents are stored if they are in the Pareto front. If another agent outperforms the old agent in the Pareto front, its network parameters are replaced with those of the new agent. Therefore, to some extent, the proposed algorithm is adjustable; it can change the bias towards both carbon emissions reduction and electricity cost optimization by altering the stored agent.

Scenario 1: Time-based scheduling for comparisons

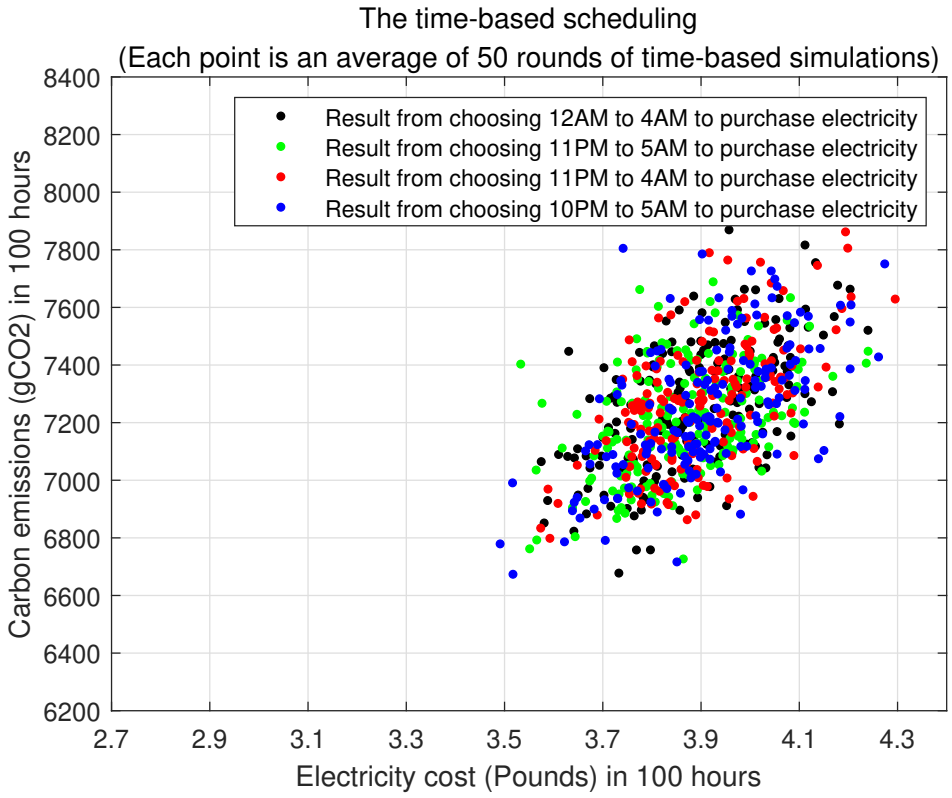

Figure 4.18: The accumulated carbon emissions and the electricity cost result from the time-based scheduling. This figure shows three simulations, choosing 12 AM-4 AM, 11 PM-5 AM, 11 PM to 4 PM and 10 PM-5 AM.

The results show that choosing 10 PM-5 AM (when the average carbon emissions are 7250 gCO_2, and the average electricity costs are 3.91 Pounds) as well as 11 PM-5 AM (when the average carbon emissions are 7216 gCO_2, and the average electricity costs are 3.87 Pounds) has better performance on both carbon emissions and electricity costs to some extent, but there are limitations. This is because the

electricity purchased at night could be insufficient for the daytime use, thus additional purchases during the day could be needed. Furthermore, not all nights have low electricity prices and carbon emissions, leading to inflexibility in this method.

Scenario 2: Multi-agent DQN for comparisons

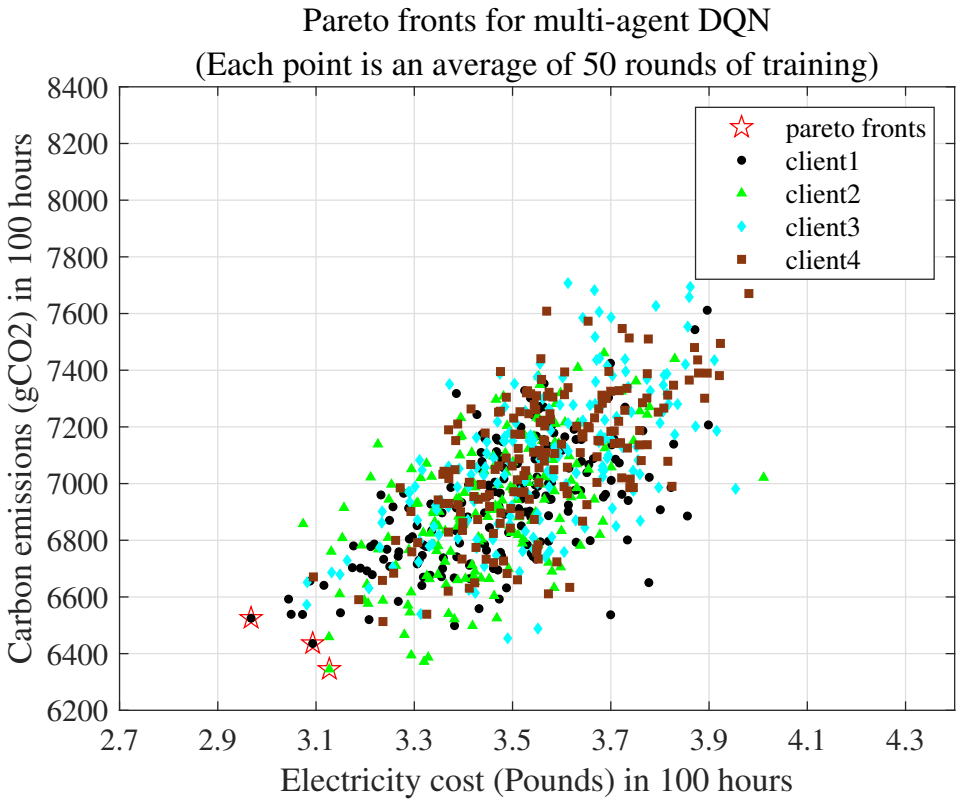

Figure 4.19: The Pareto fronts for multi-agent DQN.

The Pareto fronts of a Multi-agent DQN are shown in Fig. 4.19. In Pareto fronts, the carbon emissions vary from 6524 to 6344 gCO_2 (average 6434 gCO_2), and the electricity costs vary from 3.12 to 2.96 Pounds (average 3.05 Pounds). Compared with the average solution of 10 PM-5 AM in scenario one, it has better optimization effect, with carbon emission optimized by 11.3% and electricity costs optimized by 21.9%. Compared with the average solution of 11 PM-5 AM in scenario one, the carbon emission optimized by 10.8% and electricity costs optimized by 21.2%.

The simulations show that the optimization degree of the electricity costs exceeded the carbon emissions. A potential reason could be that the real price data was used instead of the predicted data.

Scenario 3: Proposed single DQN

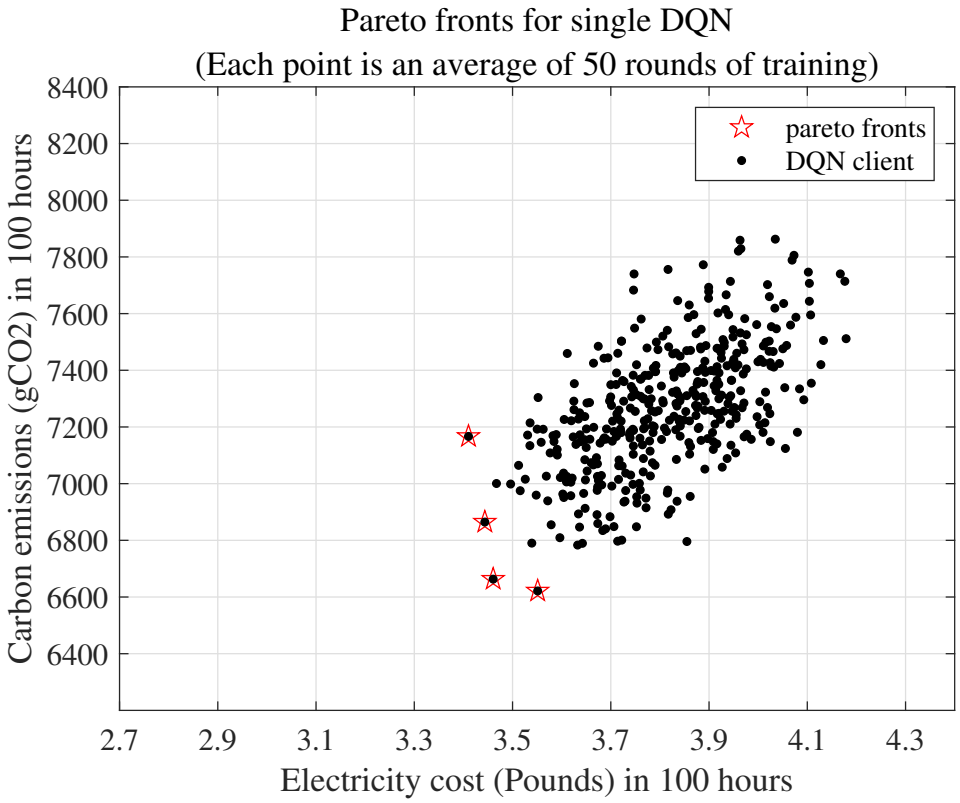

Figure 4.20: The Pareto fronts for single DQN.

The Pareto fronts of a single DQN are shown in Fig. 4.20. Each node was the average carbon emissions and electricity costs in 100 hours during 50 rounds of training iterations. The total training iterations are 50×400, which equals 20000.

In the Pareto front solutions, the average Pareto front value for electricity cost is 3.47 Pounds, and the average Pareto front value for carbon emissions is 6828 gCO_2. Compared to its own average values (electricity cost of 3.81 Pounds and carbon emissions of 7270 gCO_2), the Pareto front values optimized the electricity cost by 9% and carbon emissions by 6% further. Compared to its own highest valid values (electricity cost of 4.18 Pounds and carbon emissions of 7862 gCO_2), the Pareto front values optimized the electricity cost by 17% and carbon emissions by 13.1% further.

Compared to 11 PM-5 AM in Scheme 1, which is time-based scheduling (with optimal values of 3.87 Pounds for electricity cost and 7261 gCO_2 for carbon emissions), the optimization optimized electricity cost by 10.3%, the optimal carbon

emissions optimized by 6%. Meanwhile, the optimal solution of Single DQN also outperforms the optimal value of time-based scheduling. This indicates that the proposed Single DQN is already better than time-based scheduling, demonstrating that the proposed DQN can learn and optimize power purchases during periods of lower electricity prices as well as carbon emissions.

Scenario 4: Proposed federated DQN

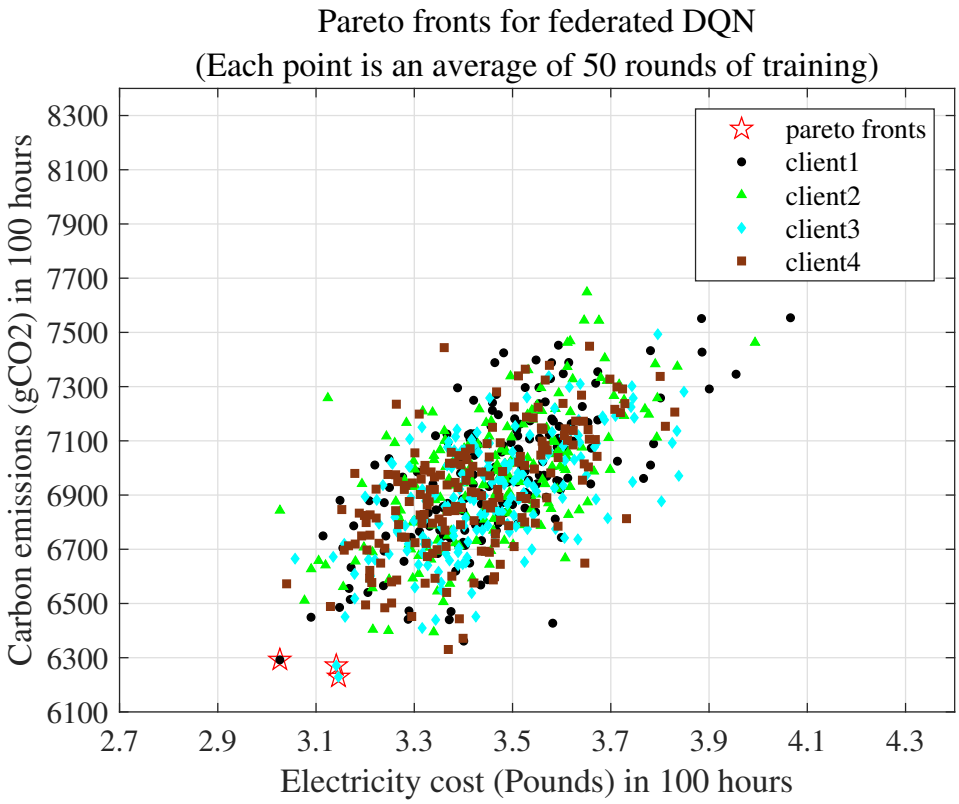

Figure 4.21: The Pareto fronts for federated DQN.

With the help of federated learning, the privacy of homes can be protected by uploading the client weights, the different colors represent different clients. 3 Pareto fronts from a federated DQN are shown in Fig. 4.21.

For different Pareto fronts, the carbon emissions and electricity costs from the leftmost Pareto front are 6291 gCO_2 and 3 Pounds (The optimal value) respectively, for the rightmost Pareto front, the values are 6229 gCO_2 (The optimal value) and 3.14 Pounds respectively. For the different DQN models from all the valid Pareto fronts in Fig. 4.21, the amplitude of the carbon emissions varies by about 1%, and

for the electricity costs, it varies by 4.4%. The average values of the Pareto front are 6263 gCO_2 and 3.1 Pounds.

Compared to average value of Scheme 1's time period scheduling, 11 PM-5 AM, the electricity cost was reduced by 13.2%, while the carbon emissions reduced by 19.9%. This indicates that the proposed federated learning algorithm can further optimize carbon emissions as well as electricity costs.

Compared with the Pareto fronts of multi-agent DQN in scenario 2, it can be found that federated learning can achieve similar performance to multi-agent learning. The carbon emissions decreased further by 2.7%, and the electricity costs increased by 1.6%. However, the multi-agent DQN has privacy leakage risks while this has been solved in the proposed federated DQN.

Compared with the Pareto fronts in single DQN in scenario 3 (the average carbon emissions were 6828 gCO_2, average electricity costs were 3.47 Pounds), the performance of federated DQN Pareto fronts increased further. The carbon emissions decreased by 8.2% and the electricity costs decreased by 10.7%. It can be concluded that the proposed DQN can still seize the opportunity to buy electricity when either the carbon emissions or the electricity prices are relatively lower than in other scenarios while still having better performance than a single DQN.

The Pareto fronts provide biases towards either electricity costs or carbon emissions optimization. The client can choose their bias towards carbon emissions or electricity costs by using different DQN models with different Pareto fronts stored in the database during the training steps according to **Algorithm 2**. When the server conducts either federated learning or multi-agent learning, it can send the parameters of a specific agent chosen by the client to the client, allowing the model to align with the client's preferences.

Overall, the proposed federated DQN can maximize the use of each client's data while at the same time protecting privacy, achieving dual-objective optimization of both the cumulative electricity costs and carbon emissions.

4.6.7 Algorithm space complexity

In the proposed architecture, the space complexity of a client is determined by:

1) DNN for Q-table approximation in DQN: This includes the online and target networks in DQN. For example, for a network structure with an input layer of 7×30, a hidden layer of 30×30, and an output layer of 30×1, the space complexity of a single network is calculated as $7 \times 30 + 30 + 30 \times 30 + 30 + 30 \times 1 + 1 = 1201$, where 7×30, 30×30 and 30×1 are the input dimension $\times$ output dimension while $+ 30$ and $+ 1$ is the dimension of the biases. Considering the target network, the overall space complexity is $1201 \times 2 = 2402$.

2) Memory Replay: Used for DQN's experience replay, its size is typically set, and the designated size determines its space complexity. For instance, if Memory Replay is set to retain 50000 data positions, the space complexity is 50000 [137].

3) Pareto front introduction: Due to the introduction of the Pareto front, additional storage is needed for networks considered part of the Pareto front. For example, if the training process retains the network models of 5 Pareto agents, the space complexity is $1201 \times 5 = 6005$.

4) Federated learning: Introducing federated learning adds communication overhead but does not increase the space complexity for an individual client [75].

5) Reinforcement learning part of DQN: The reinforcement learning part of DQN does not increase space complexity significantly; it only requires storing some reinforcement learning parameters and can be negligible [207].

6) Packet overhead: Packet overhead is a simplified measure of communication, focusing primarily on the size and number of data packets. In federated DQN, communication with the server occurs twice every 10 rounds. This includes transmitting the local client's model parameters to the server and receiving the global model parameters from the server. Therefore, the communication overhead between a client and the server over 100 rounds is twice the space complexity of the DNN for Q-table approximation in DQN, i.e., 2402.

In summary, in this simulation, the original DQN algorithm's space complexity is 52402, and after introducing the Pareto front, the algorithm's space complexity becomes 58407, representing an 11% increase (excluding packet overhead). It can be observed that the introduced space complexity is mainly determined by the size of the Memory Replay.

4.7 Conclusions

This chapter introduces the multi-objective federated scheduling algorithm to optimize carbon emissions in the home microgrid system. The contributions are summarized as follows:

- Development of a home microgrid smart energy scheduling algorithm using DQN. The LSTM is utilized for solar power and load data prediction, creating the environment for the DQN in the next time step. The operation of the DQN is based on the predicted data to minimize the accumulated carbon emissions and electricity costs during consecutive 100-hour periods. The trained DQN can acquire the best electricity purchasing opportunity when the overall carbon emissions and electricity prices are at their lowest.

- Development of a distributed structure with federated learning to enhance the efficiency of the DQN algorithm while addressing privacy concerns. In the DQN, the deep neural network parameters that approximate the Q-table are uploaded to a server, enabling federated learning. The comparison was made with time-based scheduling, multi-agent DQN and single DQN.

- Development of a Pareto front for the multi-objective DQN, comparing the trade-offs between different objectives. Each point on the Pareto front represents a DQN agent, allowing for adjustments based on specific requirements. This enables the agents to lean towards reducing either electricity costs or overall carbon emissions.

From the simulation, it can be found that:

1. From sub-section 4.6.6, it can be found that with the single DQN, the carbon emission decreased by 13.1%, while the electricity costs decreased by 17%. The proposed DQN method can seize the opportunity to buy electricity when carbon emissions or electricity prices are lower than at other times.

2. From scenarios 4 in the sub-section 4.6.6, the proposed federated method is able to capture opportunities for lower carbon emissions or lower electricity

prices than the single DQN; the carbon emissions decreased further by 8.2%, and the electricity costs decreased additionally by 10.7%. The proposed algorithm achieved a performance similar to that of the multi-agent Deep-Q network, which however, has privacy information leakage concerns. With the help of federated learning, the privacy of homes can be protected by uploading only client weights.

3. The simulations show that the degree of optimization of the electricity costs is able to exceed that of the carbon emissions. This is because real price data was used instead of only predicted data.

4. The Pareto fronts provide biases towards either electricity costs or carbon emissions optimization. To protect privacy, the clients can choose whether their bias is towards lowering carbon emissions or reducing electricity costs by using different DQN models on the Pareto fronts.

In this Chapter, federated learning is considered to protect user privacy; however, research indicates that the gradients in federated learning still contain information that may cause privacy leakage if an attack occurs. Therefore, how to measure privacy leaks in federated learning and propose recommendations for further protecting private data is a research that is required to be explored in Chapter 5.

CHAPTER 5

Privacy protection in federated home applications

While federated learning can avoid the uploading of private data, recent research indicates that there are still many risks associated with the use of federated learning [114]. Up to now, there have been three main types of privacy leakage algorithms to attack federated learning: membership inference attacks, GAN-based attacks, and gradient inversion-based attacks (such as DLG attacks). Membership inference attacks are suitable for multi-classification problems, but their success rate is often very low for binary classification or regression tasks, thus it is not the major threat when it comes to regression tasks in power systems [120]. GAN-based attacks assume that a certain client is a malicious attacker, and that this client needs to have a significantly different distribution from the other clients, making the attack conditions stringent [121]. Therefore, membership inference attacks and GAN-based attacks are not considered in this thesis.

For Gradient inversion-based attacks, if there is a controlled or inherently malicious server or client, the use of DLG (Deep Leakage from Gradients) [114] or GS (Gradient Similarity) [115] algorithms can recover the original partial training data based on the gradient differences in federated learning. In some cases, it may even accurately recover the entire training data. Compared to other attack methods,

gradient inversion-based attacks can directly compromise privacy when the training gradient data is received by the attacker, as these attacks have simpler requirements but pose a greater risk of privacy leakage. Therefore, attacks in the DLG family are chosen as the main focus of this thesis.

Within the DLG family of algorithms, GS improves upon DLG and often has better optimization effects. iDLG is suitable for classification algorithms and improves recovery accuracy, while Gradient Inversion is suitable for computer vision, like image recovery tasks. To sum up, GS is the most suitable DLG family algorithm for power scenarios.

In the previous two chapters, many scenarios have utilized federated learning [75], such as recommendation systems, PV and load forecasting, electricity scheduling, etc. The potential risk of privacy leakage is as follows:

1. The household device action recommendation system with federated learning: The household device action recommendation system uses the power consumption data for each appliance. Based on this information, the status of each appliance can be determined, and the next appliance likely to be used is able to be predicted. If a hacker gains the training data, it would be easy to discern the user's electrical usage habits, including the specific appliances usage trends and the user's absence patterns. This poses a serious threat to user privacy.

2. Load forecasting: The input for the load forecasting system is the load and voltage data from the previous three hours, and the output is the forecast for the next hour. Although load information is the overall value, it does not reveal the individual usage trends for each appliance. However, it does expose the general electricity usage habits of the user. Research indicates that non-intrusive load monitoring algorithms can be used to analysis and convert the overall load data into the usage trends for each appliance's power consumption [197] [208]. Therefore, the leakage of load data could also compromise user privacy.

3. Photovoltaic power forecasting: After converting PV power into electricity through PV panels, photovoltaic power can be considered as part of the supply

to the load. From this point of view, PV forecasting can be seen as part of load forecasting, which also has potential privacy issues.

4. Scheduling algorithm: The scheduling algorithm is based on DQN and was introduced in Chapter 4, in which the environmental data, such as the load data, remaining battery capacity, photovoltaic output, electricity price, carbon emissions, etc. are used. The outputs are the Q-values of the DQN algorithm. This also includes privacy information like load data and photovoltaic output. Electricity prices and carbon emissions data are public; therefore, they do not pose a privacy risk. Since the battery is assumed to be the communal property, there is no privacy leakage under the GS attack. Hence, the privacy leakage in DQN aligns with scenarios 1 and 2.

According to the analysis above, the major privacy leakage application scenarios are 1) The household appliance action recommendation systems and 2) The load forecasting system. This chapter proposes an algorithm to measure the potential risk of privacy leakage in neural networks. Simulations are conducted in the above household applications, and suggestions for avoiding privacy leakage at the household level are provided based on the theoretical analysis and simulation results.

The proposed system architecture, SAFE-Home (Smart Applications Federated Learning with Emphasis on Home Privacy), can be described in Fig. 5.1.

The proposed system comprises home metering units (clients) and a management system (federated learning server) to perform the SAFE-Home algorithm. The clients collect household metering data, where voltage and load data, triggering time, and location of appliances are collected. The server and clients perform federated learning for 1) load and voltage forecasting and 2) household devices action recommendation by uploading the model parameters of the clients. Then, the federated learning data is used and sent to the Attack module, which executes GS attacks targeting the gradient differences in federated learning. Subsequently, the recovered data is paired with the original data, and the Kendall coefficient is used to calculate their similarity [209]. The attacked data and the similarity results are then sent to the Explainable module to identify the key factors causing privacy leakage. Finally, based on the results of the explanation, privacy protection advice is provided

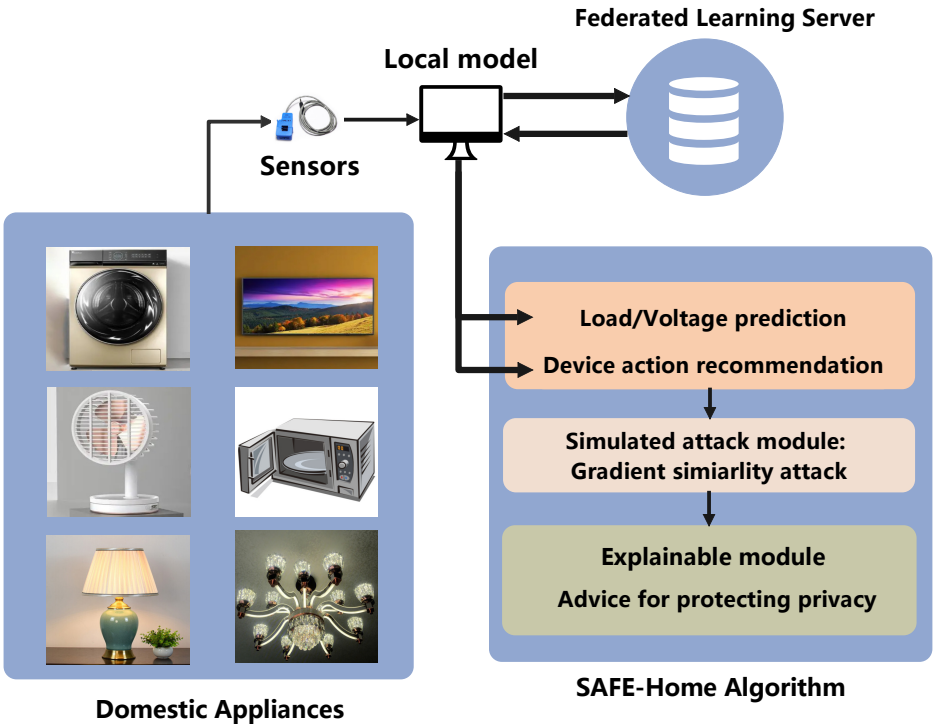

Figure 5.1: An illustration of a privacy leakage measurement structure, SAFE-Home. The federated home application scenario consisting of home metering units (client) and an management system (a federated learning server). The proposed algorithm in SAFE-Home is used to measure the privacy leakage in the gradient.

to households by analyzing the key factors in federated learning that may lead to privacy leakage.

5.1 Household applications and attack module

This section introduces foundational work from Chapters 3 and 4, encompassing two smart home application scenarios: load and voltage prediction and household device action recommendation, shown in Fig. 5.2 with structure changed. Then, this section outlines how these scenarios employ federated learning and describes the GS attack based on their federated learning gradients.

5.1.1 Load and voltage forecasting

The first application aims to predict the average power and voltage for the next time period. This application scenario is derived from Chapter 4, where load prediction

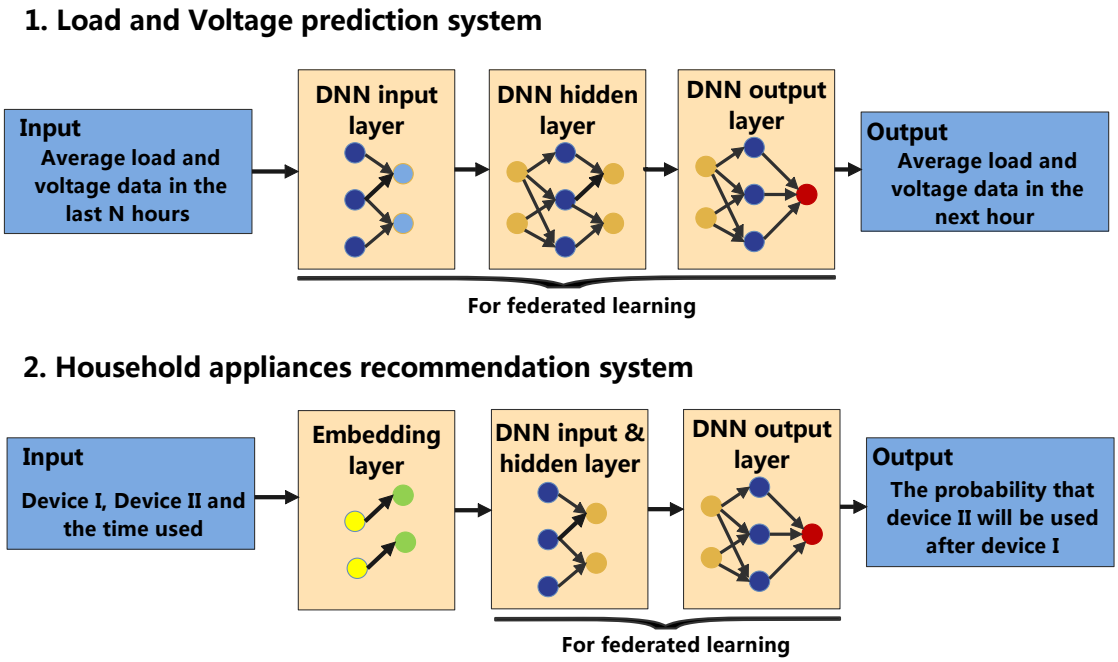

Figure 5.2: An illustration of the basic structures in household smart applications, including: 1. The load and voltage forecasting, and 2. The household devices action recommendation.

is performed to create the state space for the DQN algorithm [137] in the next time step. The difference lies in the inclusion of voltage prediction in this chapter, and the DNN structure [129] is used instead of the LSTM. The structure is shown in the upper half of Fig. 5.2.

The power and voltage datasets are organized into time series data and then normalized. Considering a neural network composed of fully connected layers, with inputs including the average load and voltage of the first few time periods and outputs the average load and voltage of the following time period.

5.1.2 Household devices action recommendation

The second application aims to generate recommendations for the next household appliance actions. This application scenario is from Chapter 3. Among them, the knowledge graph construction and training methods are completely identical in Chapter 3. The difference is that a DNN [129] is used instead of the KGAT layer while retaining the constructed graph dataset and embedding layer. The structure is shown in the lower half of Fig. 5.2.

The overall structure of the recommender system is as follows: Household appliance data is integrated into a knowledge graph, encompassing information such as appliance indexes, the times that the appliance is used, locations, the sequence of appliance usage, rated power, and user expected satisfaction coefficient. Initially, the data is encoded into embeddings. Then, the embeddings of two devices and the corresponding usage time embeddings are concatenated and input into a DNN layer. The output is between 0 and 1, indicating the likelihood of the two appliances being used sequentially.

5.1.3 Federated learning with applications

For both scenarios, federated learning is conducted on models from multiple households to protect user privacy. Each home uploads the weights during learning and executes the federated learning algorithm on the server side. In the federated learning model of both scenarios, the batch gradient $\nabla L(w(n))$ after training can be calculated using the training data and label [75]. As an example, the mini-batch data and labels are considered as

$$\nabla L(w(n)) = \frac{\partial l(Flr(mb, w(n)), m)}{\partial w(n)}, \tag{5.1}$$

where $l(\cdot)$ is the loss function like mean squared error loss. Flr could be the load and voltage forecasting or household devices action recommendation model mentioned above. mb is the mini-batched input, including the historical data in the past time intervals. $w(n)$ is the model parameter, including weights and biases during the time interval n. m are mini-batched labels.

The local training part of the federated learning is the main focus of the simulation, the period for the client i to conduct local training with the local data after receiving the global model from the server [75].

$$w_i(n+1) = w_i(n) - \eta_{\mathrm{FL}} \nabla L(w_i(n)), \tag{5.2}$$

where i represents the i-th client and η_{FL} is the learning rate. After each client local update, the time interval plus one, the time increases from n to $n+1$.

Then the client i sends k_{FL}-turn trained parameters $w(n+k_{\mathrm{FL}})$ of the model to the server. If all weights of clients are received, the server calculates the average weight of each client and then sends the averaged model to each client. The averaging process is described by [75].

$$w_{\mathrm{averaged}}(n+k_{\mathrm{FL}}) = \frac{1}{I}\sum_{i=1}^{I} w_i(n+k_{\mathrm{FL}}), \tag{5.3}$$

where $w_{\mathrm{averaged}}(n+k_{\mathrm{FL}})$ is the averaged global model and I is the number of clients.

5.1.4 Attack module

While federated learning shares the gradient parameters of the network, it is assumed that a malicious client or server will record the gradient difference between two updates and perform GS gradient inversion from (5.4) to (5.5) to obtain potential privacy input data, which can be described as follows [114] [115].

$$\nabla L(w_{\mathbb{D}}(n)) = \frac{\partial l(Flr(mb_{\mathbb{D}}, w(n), m_{\mathbb{D}})}{\partial w(n)}\ , \tag{5.4}$$

$$\{mb_{\mathbb{D}*}, m_{\mathbb{D}*}\} = \underset{mb_{\mathbb{D}}, m_{\mathbb{D}}}{\operatorname{argmin}} \frac{\sum\left(\nabla L(w(n))\nabla L(w_{\mathbb{D}}(n+k_{\mathrm{FL}}))\right)}{\sqrt{\sum\left(\nabla L(w(n))\right)^2}\sqrt{\sum\left(\nabla L(w_{\mathbb{D}}(n+k_{\mathrm{FL}}))\right)^2}}, \tag{5.5}$$

where subscript $\mathbb{D}$ means the randomly initialized. $mb_{\mathbb{D}}$ represents the randomly initialized data, $m_{\mathbb{D}}$ represent the randomly initialized labels. The model parameters $w_{\mathbb{D}}(n)$ are calculated with $mb_{\mathbb{D}}$ and $m_{\mathbb{D}}$. The $\nabla L(w_{\mathbb{D}}(n))$ are the gradients generated by $mb_{\mathbb{D}}$, $m_{\mathbb{D}}$ and $w_{\mathbb{D}}(n)$.

Similar with (5.2) and (5.3), after k_{FL} rounds training, the weights $w_{\mathbb{D}}(n+k_{\mathrm{FL}})$ can be gotten. With that, the gradient $\nabla L(w_{\mathbb{D}}(n+k_{\mathrm{FL}}))$ can be calculated. The randomly initialized data $mb_{\mathbb{D}}$ and labels $m_{\mathbb{D}}$ can iteratively approach real values mb and m by optimizing the cosine similarity between $\nabla L(w(n))$ and $\nabla L(w_{\mathbb{D}}(n+k_{\mathrm{FL}}))$ in (5.5), this is because GS algorithm indicates that the same inputs produce same gradients. $\{mb_{\mathbb{D}*}, m_{\mathbb{D}*}\}$ are the assemble of the optimized (recovered) data and labels, which are closer to the real data and labels after optimization.

Although the same inputs produce the same gradients, when performing a gradient inversion using a GS attack, the same gradients do not necessarily recover the same inputs. This is because the gradients have a high dimensionality, and the same gradients can correspond to multiple different inputs. These inputs can produce the same gradients through different paths.

The attack process is recorded with detailed data, including training batch size, iterations and network structures. The training batch size here refers to how many training samples are backpropagation updated at a time, and the process of updating parameters using batch size data is a training iteration. For training iterations, this chapter specifically refers to local training iteration on the client side rather than global training. Network structures refer to the structure of layers in a network, including fully connected layers and their size information (It is also called the 'hidden size' in this thesis), activation functions, etc. For load and voltage prediction, the attack attempts to recover the original training data, while the attack on device action recommendation attempts to recover the input embedding, thereby inverting the original data. The data collected during the final attack is sent to the explainable module in section 5.2.4 and 5.2.5 to measure the degree of privacy leakage and locate vulnerable privacy data.

5.2 Proposed SAFE-home algorithm

This section presents the SAFE-Home algorithm, including the proposed pairing principle and the explainable algorithm after GS attacks. It includes five steps:

1. Attack the existing federated learning networks with the GS algorithm.

2. Pair the attacked data with the original data.

3. Calculate the similarity between the recovered data and the original data with the Kendall coefficient.

4. Provide preliminary explanations, pinpointing the key factors leading to privacy leakage with an LSTM.

5. Offer in-depth explanations using a weighted approach with the explainable machine learning algorithm.

Components 4 and 5 form a bi-layer explanatory system designed to predict the probability of data leakage during data transmission in federated smart home applications given various batch sizes, training iterations, hidden neuron sizes and activation functions. The overall system structure is shown from Fig. 5.3 to 5.8, representing steps 1-5.

Each section, serving as a module, possesses decoupling properties and can be replaced by similar algorithms or measurement methods.

5.2.1 Attacks for the federated learning

The gradient similarity attack

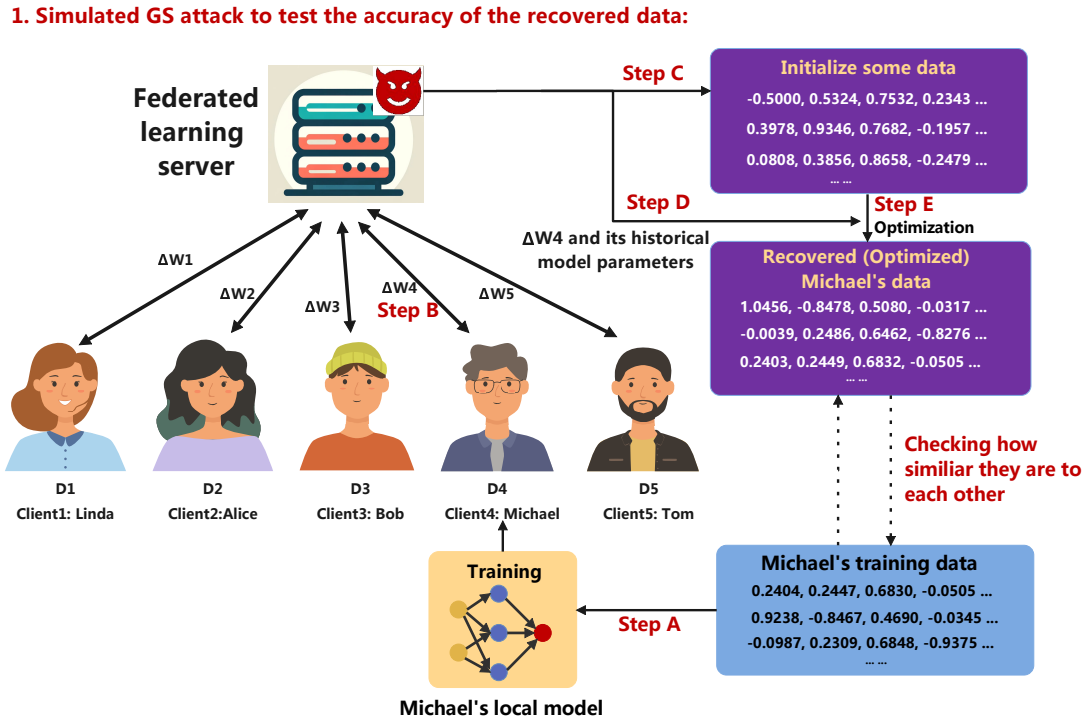

Figure 5.3: The GS attack for household applications, after the data is recovered, the recovered (optimized) data from Michael is similar to Michael's original training data, leading to privacy leakage.

To prevent privacy leakage, federated learning keeps sensitive data locally and only uploads network parameters. Therefore, if privacy leakage occurs under federated learning, it is mainly associated with various attack algorithms.

Assuming the malicious attacker could launch an attack, simulating the attack in a lab environment before it happens is essential because it is allowed to view the attack from the opponent's perspective and measure the privacy leakage based on the obtained information. The attack in this section can be understood in two ways: 1. From an adversary's perspective, simulating the attack and measuring the attack's potential to cause damage under different conditions will provide reference recommendations for future defence. 2. From the defender's perspective, it is essential to be able to do some predicational attacks to obtain the attack results and measure the privacy leakage, thus future attacks can be prevented.

In this Chapter, the GS attack in Section 5.1.4 is chosen and simulated to recover the data [115]. The GS attack can be replaced with other attack algorithms as long as they can compromise the privacy of federated learning, such as membership inference attacks [120] or GAN-based attacks [121]. Both DLG [114], and GS are able to reconstruct the original training data when compared with different attack algorithms, making them considered to be the most threatening type of attack against federated learning. Therefore, the GS algorithm has been chosen as the attack algorithm in this chapter.

The GS attack occurs when the server sends the complete model to the client and then receives back the new model parameters from the client. The attacker is assumed to be the server, which can attack by comparing the model parameters sent to the client with those received back from the client.

As shown in Fig. 5.3, five individuals, Alice, Linda, Bob, Michael, and Tom, are collaborating in a federated learning system. Next, while Michael is focused on the local training process, a malicious attacker on the server attacks Michael with a GS attack. The attacking process was described in sub-section 5.1.4, which can be divided into the following steps.

1. Michael trains his local neural network with original data, such as original embeddings, voltage, and load data. The network parameters and structures of the local training model, including the batch sizes, training iterations, hidden sizes (the size of the hidden neurons of the federated learning), activation functions, super parameters such as the learning rate, etc., are known by

everyone in federated learning, as it is required to perform federated learning and all this data is stored.

2. After local training, the gradients and outputs (the predicted values) can be obtained by Michael, as well as a trained local model. In federated learning, it is necessary to send the trained local model (or its gradient gaps compared with the global model) to the server. It should be noted that a malicious server might have a record of the model that was sent by the client from their last interaction. From this, the parameter differences between the two versions of the model can be deduced. In deep learning algorithms, knowing any two of the value factors (1. model parameter differences, 2. the learning rate, and 3. the model training gradient) allows for determining the third. Therefore, a malicious attacker in the server could infer the gradient from the private training data and consequently perform a GS attack in the future steps.

3. The malicious server randomly generates some data, the same size as Michael's training data, and inputs it into the global network that server recorded last time for training (this was sent to Michael for local training). Whether to use random data depends on whether the server has prior training data from the attacked client. If it does, using the prior data instead of random data often results in a more effective attack. The random data can be optimized to approximate the private training data by using the GS attack.

4. The second set of outputs and gradients are obtained, based on the randomly initialized (dummy) data. The L-BFGS optimizer (or the ADAM optimizer) is used on the dummy data to minimize the cosine similarity between the two sets of gradients until the number of optimization rounds required to complete the training is reached or a situation arises where the cosine similarity between the gradients becomes extremely small [210].

5. The optimized dummy data results in recovering the embeddings, voltage, and load data which are highly correlated with the original data.

Steps 1 and 2 simulate the client training process locally. The remaining steps

involve the server attacking and obtaining the client's original data. Subsequently, the original and recovered data proceed to the next pairing step.

Computational complexity

When performing a GS attack, assuming the parameter differences and learning rate of the federated model are known, it is necessary to first calculate the gradients, which is the reverse operation of equation 2.1. The time complexity for calculating gradients is $\mathbb{O}(1)$ for simple fundamental calculations from the model parameter gaps and the learning rate.

In addition, the GS algorithm optimizes the input training data based on gradients, and its computational complexity depends on both the complexity of the optimization algorithm and the number of optimization rounds. Assuming the Adam optimization algorithm is used, the computational complexity per round is $\mathbb{O}(n)$ (Adam optimization includes computing the first and second moment estimates for each parameter, applying bias correction to these estimates, and then updating the parameters) [211]. The number of rounds required for convergence is uncertain, and if it converges after adt rounds, the total computational complexity would be $\mathbb{O}(adt \times n)$. Therefore, if the optimization converges slowly, adt could be large, making the optimization process both time and computationally consuming.

In summary, the overall computational complexity is $\mathbb{O}(1)+\mathbb{O}(adt \times n)$, with the latter (the complexity of the optimization process) being the dominant factor.

5.2.2 Pairing principle

After the GS attack, the real input and recovered data are paired in SAFE-Home according to the minimum L2 distance. Observing the results, multiple recovered data is believed to match one original data.

The steps for data pairing are as follows: the L2 distance from one data point in a batch to all original data points is iteratively calculated. Choosing the minimum absolute values in all the pairs for matching. The L2 distance here can be replaced with other metrics, such as cosine similarity or mutual information. When the batch size is large, the L2 distance is the method with the lowest computational

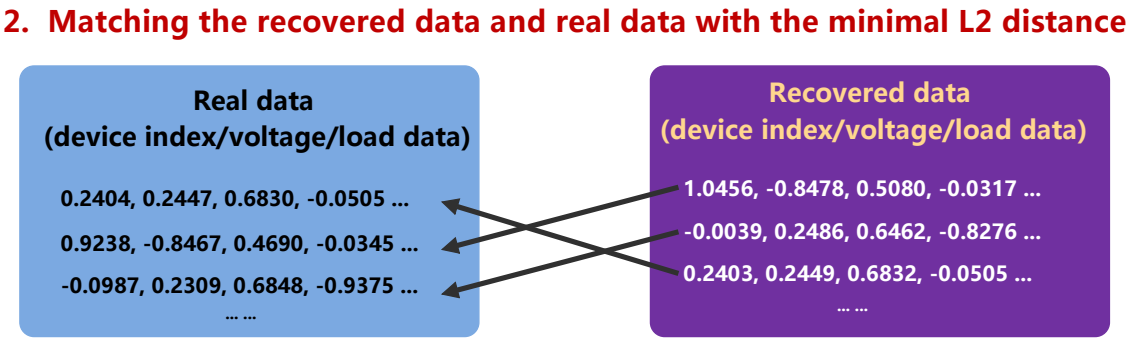

Figure 5.4: Pairing the original data with the recovered data. The recovered data is compared with all the original data within the batch size by calculating the L2 distance, and the smallest L2 distance is taken as the criterion for pairing.

complexity. Therefore, L2 distance is used here.

For example, in Fig 5.4, the first data on the left, 0.2404, 0.2447, 0.6830, -0.0505... and the data on the right, 0.2403, 0.2449, 0.6832, -0.0505... are successfully paired, because they have the smallest L2 distance in calculations.

5.2.3 Calculating similarity

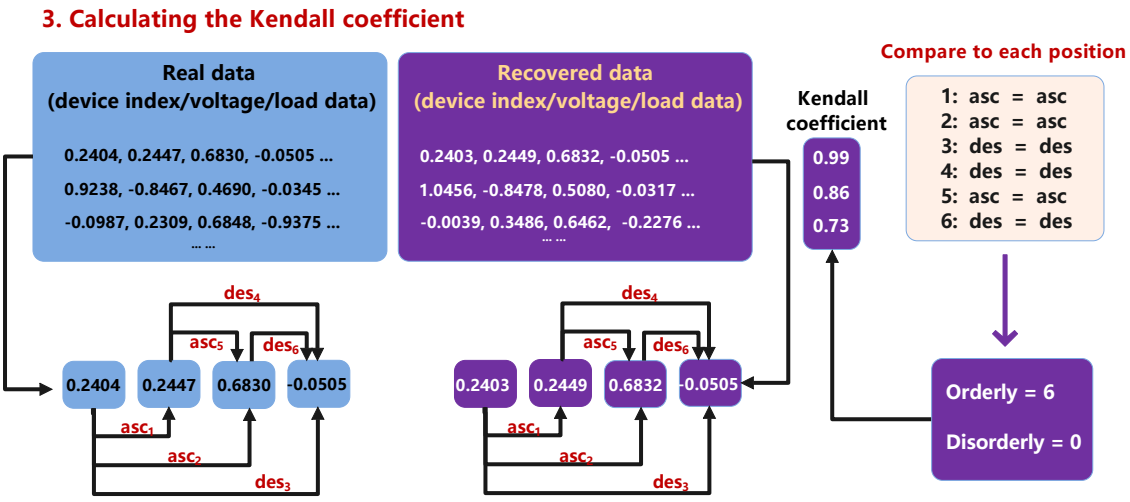

Figure 5.5: Calculating the Kendall coefficient between the original data and the recovered data. The Kendall coefficient can measure the correlation between the recovered data and the original data, especially reflecting the trend of changes over time in the load and voltage data.

Three kinds of metrics can be considered to compare the accuracy of data recovery after attacks. The first possible choice is distance-based methods, such as L2 distance and cosine similarity, which reflect the difference between the original

and recovered data. The second possible choice is that mutual information reflects the view of information theory [212]; the probability distribution between two variables is used to measure the correlations. However, for regression problems under gradient inversion algorithms, the changing trend of the variables can also be considered—for example, the trend of the accumulated load from the last few hours to the next hour. Spearman and Kendall coefficients can be used to make correlation analyses of two ordered variables, so they are more suitable to be the metrics for the forecasting problem in this case [213]. According to this feature, the correlation coefficients are chosen as privacy leakage metrics for federated home applications. In the simulation of SAFE-Home, the Kendall coefficient was used to determine the correlation between the paired original and recovered data, which is shown as [209]

$$\text{Kendall} = \frac{\beta - \psi}{length(length-1)/2}, \tag{5.6}$$

where β and ψ are the orderly and disorderly rankings of the recovered load and voltage data or embeddings, and $length$ is the number of statistical objects. For the prediction algorithm, the sequence length is twice the length of the time series for voltage and load. For the recommendation algorithm, the sequence length is the sum of the lengths of the input embeddings. The result falls into [-1,1], where 1 means positive correlation, -1 means negative correlation, and 0 means irrelevant.

More specifically, Fig. 5.5 depicts an example of where the original and recovered data are. Assuming that only the first four data points are taken to calculate Kendall's coefficient, the steps are as follows:

The data are paired in combinations, such as the first data with the second, the first with the third, the first with the fourth, and so on, until the pairing of the third data with the fourth. The ascending or descending relationship between the original and the recovered data is recorded for each data pair. The number of pairs where the original and recovered data are consistent is noted as β, and the number of pairs where they are inconsistent is noted as ψ. In Fig. 5.5, β=6, ψ=0. Assuming the length of the data is 4, so $length$=4. Afterwards, formula (5.6) can be used to calculate the Kendall coefficient between the original and recovered data. Under

the premise of considering only the first four data points in the graph, the Kendall coefficient is 1 between them.

This system assesses the similarity between the recovered and original data based on input data, data batch size, hidden size, and training iterations. The evaluation uses the Kendall coefficient, which indicates the degree of similarity between the two datasets, allowing for determining the extent of privacy leakage. If the Kendall coefficient is significantly high (near 1), it indicates a complete privacy leakage.

5.2.4 Explainable layer I

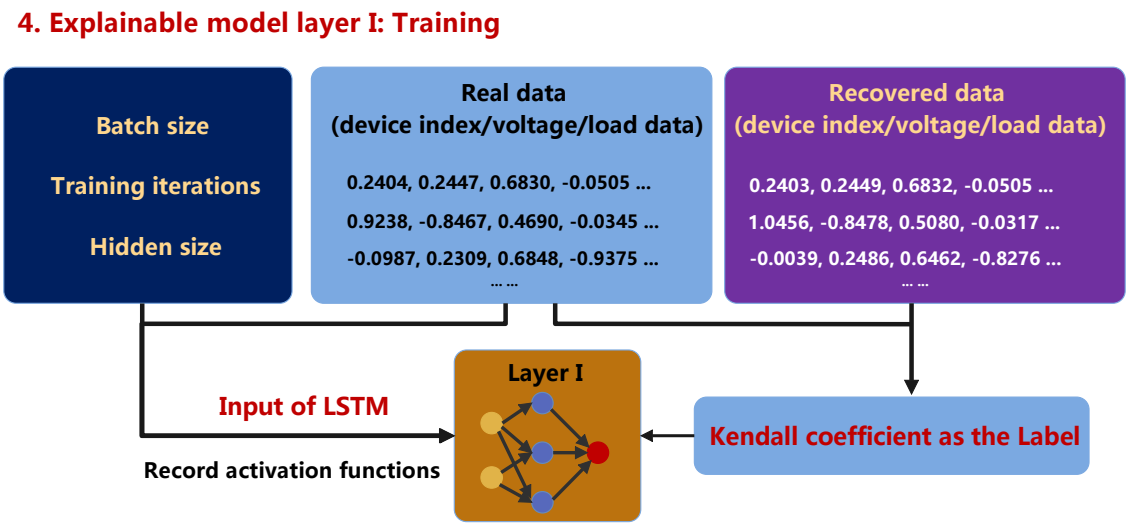

Figure 5.6: Training in explainable layer I. Using an LSTM model to fit and predict the degree of privacy leakage. The LSTM model is trained using the number of batch sizes, training iterations, hidden sizes and the original data as input.

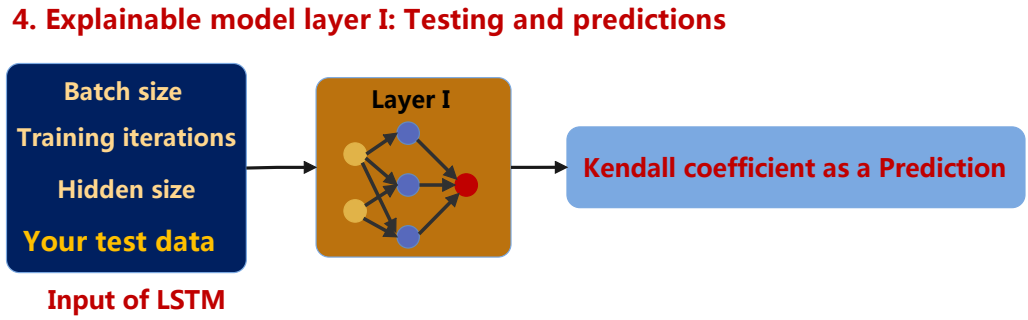

Figure 5.7: Using and testing in explainable layer I. Given a number of batch size, training iteration, hidden size, and specific test data, a well-trained network produces a corresponding Kendall coefficient from a trained LSTM network. The larger the coefficient, the more it indicates privacy leakage.

In the first layer of measuring privacy leakage in SAFE-Home, it is necessary to establish a prediction system to predict privacy leakage. Due to the unique long-term memory capabilities of LSTM, it is well-suited as a predictive algorithm, so an LSTM architecture is employed in this layer, estimating the privacy leakage level under GS attacks based on historical data [214]. The input and output of the LSTM are shown in formula (5.7) and (5.8).

$$\text{Input} = [bch,\ ig,\ hsi,\ mb,\ m], \tag{5.7}$$

$$\text{Output} = \text{Kendall}_{\text{prediction}}(mb_{\mathbb{D}*},\ m_{\mathbb{D}*},\ mb,\ m), \tag{5.8}$$

where the input consists of batch size bch, training iterations ig, hidden-layer size hsi, raw data mb, and label m. The labels are the Kendall coefficients between the original and recovered data. A global perspective of the privacy leakage can be obtained by fitting this network, as shown in Fig. 5.6. The activation function is not used as an input to the LSTM but rather as a marker of the network. This means it's assumed that the network under attack only uses one type of activation function, and it's recorded. As shown in Fig 5.7, after the training is completed, during the testing or usage phase, the well-trained network can predict the level of privacy leakage when given input parameters like batch size, training iteration, hidden-layer size, and original data. The range of the predicted values is restricted between 0 and 1 (since there is a possibility that the Kendall coefficient can be less than 0, during training, Kendall coefficients less than 0 are set to 0), where 0 represents no privacy leakage, and 1 represents complete privacy leakage.

5.2.5 Explainable layer II

The second layer is built on layer I and is used to precisely identify the factors and ranges that positively or negatively impact privacy leakage. This layer's algorithm is based on the explainable machine learning algorithm LIME [189]. There are many explainable machine learning algorithms, such as LIME, Shapley, and Grad-CAM. The reason for choosing LIME is its model-agnostic nature and its ease of

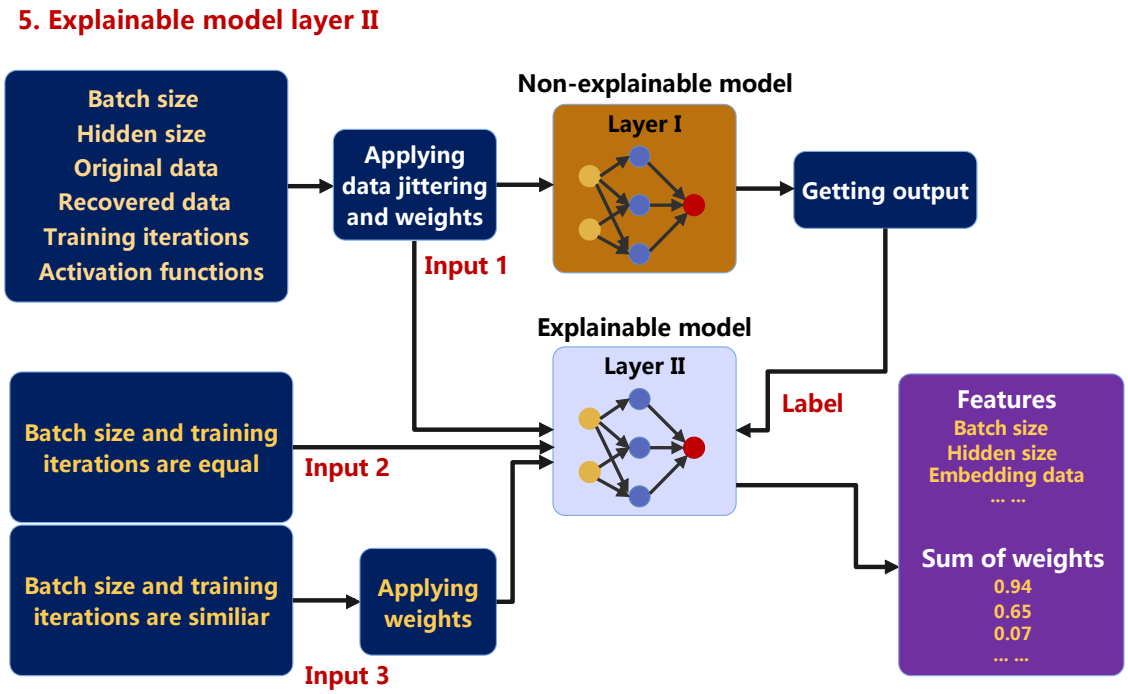

Figure 5.8: On top of the first layer, the SAFE-Home framework incorporates an improved explainable machine learning algorithm to further explain the factors affecting privacy. In the purple box, the 'sum of weight' represents the explanation for each type of feature. This illustrates how these features affect privacy leakage.

customization. The algorithm proposed here integrates home application scenarios and introduces the following improvements based on LIME:

In LIME, only one point is selected for data perturbation and explanation, as described in the formula (2.20). This method relies on fitting an explainable model with the input's perturbation and the neural network's outputs. Since voltage, load consumption, and household device recommendation system data are normalized or embedded, and the data range fluctuates relatively smaller than other scenarios, it can be considered the neighbourhood concept in the LIME algorithm. In the SAFE-Home, the neural network comes from step 4 in the section 5.2.4, and its fitted data comes from step 3 in the section 5.2.3. As the attack steps in step 3 are repeated multiple times, the data from step 3 can be directly used as 'perturbed' data. For this, a sampling rule is designed:

1) For situations where batch size and training iterations are equal, the data is

directly input to LIME as perturbed data without weighting.

$$L(f_{nim}, f_{im})_{\text{equal}} = \sum_{i=1}^{N_{exp}} (f_{nim}(ze_i^*) - f_{im}(ze_i^*)), \tag{5.9}$$

where the data in this group is expressed as ze^*.

2) For situations where batch size and training iterations are similar, weighted data is input to LIME.

$$L(f_{nim}, f_{im})_{\text{similiar}} = \sum_{i=1}^{N_{exp}} (w_i \cdot f_{nim}(zs_i^*) - f_{im}(zs_i^*)), \tag{5.10}$$

where the data in this group is expressed as zs^*.

3) Perturb the original data according to the LIME algorithm and input it as exploration data with weighting to LIME.

After improvements, the loss function is expressed as $L(f_{nim}, f_{im})_{\text{proposed}}$ in (5.11).

$$L(f_{nim}, f_{im})_{\text{proposed}} = L_{\text{equal}} + L_{\text{similiar}} + L_{\text{lime}} \ . \tag{5.11}$$

This approach reduces the computational resources consumed by sampling and forward propagation and theoretically explains results as more stable and accurate because the real data are used instead of the fitted data. Therefore, the explanation results are no longer specific to a single data point but rather a group of data points. Additionally, the explanation for each type of data can be summed, and by considering the explanation of all data comprehensively, a global data explanation is obtained.

Finally, a comprehensive comparison of the results of layers I and II can be conducted to understand the privacy leakage situation from different perspectives.

5.2.6 SAFE-Home algorithm

GS attacks threaten federated smart home applications due to their input inference ability. An algorithm was proposed to measure the degree of privacy leakage for smart home applications under different conditions.

The proposed SAFE-Home algorithm is shown in **Algorithm 3**: Lines 2 to 13 are the GS attacks and the collection of attack data. Lines 7 to 11 describe the attack and use the Kendall coefficient to measure the similarity between the original data (line 9), repeating the attack N_{rep} time for collecting data. In line 10, The training iteration i, batch size bch, activation function ϕ, the size of the hidden neuron (hidden size) hsi, the original data org_{bch}, the recovered data rec_{bch}, and the similarity between them $Similiarity_{i,bch}$ calculated from the Kendall coefficient is collected in $dataset$.

After attacking smart home applications, another neural network (in sub-section 5.2.4, an LSTM) is established in line 14 to fit the attack results, with input as batch size, training iterations, hidden size, and the label as the Kendall coefficient between the recovered and original data, the activation function used is also recorded. With the trained $network$, the privacy leakage can be measured with the given inputs. The detailed process is described in sub-section 5.2.4, the first layer of the proposed explanation algorithm.

Afterwards, based on the idea of explainable machine learning [166] in the sub-section 5.2.5, from lines 15 to 19, the aim is to locate the input data that is crucial for output and to determine which intervals of the input will have a positive or negative impact on the output. A set of data points is selected for the jitter according to the formula (5.9) to (5.11) according to the design of the loss function, and then a new network $local\_model$ is created to fit the impact of the jitter input. With the same thought, it can be determined which factors will exacerbate privacy leakage and which kind of data is vulnerable to privacy leakage.

5.3 Theoretical analysis

This section aims to analyse the factors that will influence the performance of the gradient inversion in DNN.

For the a-th layer of the DNN and this layer's input y_{a-1}^{bch}, (5.12) describes the fully connected layer and its output x_a^{bch}, which then gives this a-th layer output y_a^{bch} after the activation function $\Phi_a(\cdot)$ in (5.13) [129].

Algorithm 3: SAFE-Home algorithm for privacy leakage measurement

Input: The basic model *model*, including load and voltage prediction or device action recommendation system.
for *Training iterations i from 1 to* N_{iter} **do**
    Load the basic model $model_i$ at iteration i.
    Get activation function Φ and the hidden size hsi from $model_i$.
    for *Batch size bch from 1 to* N_{bch} **do**
        Load the recording of the training data org_{bch}.
        for *Repeat time from 1 to* N_{rep} **do**
            Perform the GS algorithm and get recovered data rec_{bch}.
            Get $Similiarity_{i,bch}$ = Kendall(org_{bch}, rec_{bch})
            Save $dataset = model_i, \Phi, i, bch, hsi, org_{bch}, rec_{bch}, Similiarity_{i,bch}$.
        end for
    end for
end for
Initialize a neural network *network*. Train *network* with *dataset*, that is $network(i, bch, hsi, \Phi, org_{bch}) = Similiarity_{i,bch}$.
Pick data in *dataset* according to (5.9) and (5.10) for further explanation. Disturbing data to obtain a dataset *disturb*.
Make predictions *model_predictions* of *disturb* via $model\_predictions = network.predict(disturb)$.
Compute distances *distances* from *original_instance* to *perturbed_samples*.
Train local model *local_model* with $perturbed\_samples, model\_predictions, distances$.
Get the explanation from the observations of *local_model*.
Output: *local_model*.

$$x_a^{bch} = W_a y_{a-1}^{bch} + b_a \ , \tag{5.12}$$

$$y_a^{bch} = \Phi_a(x_a^{bch}) \ , \tag{5.13}$$

where W_a is the weights of the layer a, Φ is the activate functions such as sigmoid or relu. It is assumed that x_a and y_{a-1} are mini-batched data, donated as *bch*.

With the help of gradient optimizers like ADAM [211] or L-BFGS [215], gradient inversion algorithms like DLG [114] and GS [115] can be performed. The back-propagation process of ADAM when optimizing the gradients is focused, firstly, the

loss function of back-propagation is defined as [129]

$$l = \frac{1}{2}(y_{\text{final}}^{bch} - m)^2 = \frac{1}{2}(\Phi_{\text{final}}(x_{\text{final}}^{bch}) - m)^2 \ , \tag{5.14}$$

where m is the mini-batched label, x_{final}, y_{final} and Φ_{final} means they are from the final layer of the deep neural network.

In the back-propagation, the derivative of the total loss l with respect to weights W_a and biases b_a can be obtained via chain rules.

$$\frac{\partial l}{\partial W_a} = \sum_{bch} \frac{\partial l}{\partial x_a^{bch}} \frac{\partial x_a^{bch}}{\partial W_a} = \sum_{bch} \frac{\partial l}{\partial x_a^{bch}} (y_{a-1}^{bch})^{\mathrm{T}} \ , \tag{5.15}$$

$$\frac{\partial l}{\partial b_a} = \sum_{bch} \frac{\partial l}{\partial x_a^{bch}} \frac{\partial x_a^{bch}}{\partial b_a} = \sum_{bch} \frac{\partial l}{\partial x_a^{bch}} \ , \tag{5.16}$$

where (5.15) and (5.16) are steps for back-propagation, using variables in (5.12) for derivations. The transpose T comes from the dimensional matching of matrix multiplication, where W is a column vector. It is observed that $\frac{\partial l}{\partial W_a}$ and $\frac{\partial l}{\partial b_a}$ depends on $\frac{\partial l}{\partial x_a^{bch}}$. A basic requirement of back-propagation is that the gradient cannot vanish (zero). So it is required that $\frac{\partial l}{\partial x_a^{bch}}$ is none zero. The representation of $\frac{\partial l}{\partial x_{a-1}^{bch}}$ can be obtained from the chain rule.

$$\begin{aligned} \sum_{bch} (\frac{\partial l}{\partial x_{\text{final}}^{bch}}) &= \sum_{bch} \frac{\partial l}{\partial y_{\text{final}}^{bch}} \frac{\partial y_{\text{final}}^{bch}}{\partial x_{\text{final}}^{bch}} \\ &= \sum_{bch} (y_{\text{final}}^{bch} - m) \cdot \frac{\partial \Phi_{\text{final}}(x_{\text{final}}^{bch})}{\partial x_{\text{final}}^{bch}} \ , \end{aligned} \tag{5.17}$$

$$\begin{aligned} \sum_{bch} (\frac{\partial l}{\partial x_{a-1}^{bch}})_{\text{middle}} &= \sum_{bch} \frac{\partial l}{\partial x_a^{bch}} \frac{\partial x_a^{bch}}{\partial y_{a-1}^{bch}} \frac{\partial y_{a-1}^{bch}}{\partial x_{a-1}^{bch}} \\ &= \sum_{bch} (W_a)^{\mathrm{T}} \frac{\partial l}{\partial x_a^{bch}} \cdot \frac{\partial \Phi_{a-1}(x_{a-1}^{bch})}{\partial x_{a-1}^{bch}} \ , \end{aligned} \tag{5.18}$$

where (5.17) is the formula for the final layer (5.14) while (5.18) is for the middle layers (5.12), using (5.13) for the equality substitution of $\frac{\partial y_{a-1}^{bch}}{\partial x_{a-1}^{bch}}$. The difference between them is that the loss of the last layer is the mean squared root, and the loss in the middle layer $a-1$ is conveyed by layer a. In (5.18), there exists scalar product

because $\frac{\partial y_{a-1}^{bch}}{\partial x_{a-1}^{bch}}$ is the dimensional derivation. For (5.17) the prerequisite of $\frac{\partial l}{\partial x_{\text{final}}^{bch}}$ not be zero is $(y_{\text{final}} - z)$ or $\partial \Phi_{\text{final}}(\text{x}_{\text{final}})$ not zero, which means that the data cannot produce enough gradients (over-fitting) or the chosen of some activation functions. For (5.18), the reason that the gradient disappears is the activation functions only.

The impact of gradients can be reflected in the batch size, training iterations, and hidden size. For batch size, training with a larger batch can lead to gradients overlapping, where the positive or negative gradients generated by different data may cancel each other out. Therefore, increasing the batch size can make GS gradient attacks more difficult. Regarding training iterations, a larger number of training iterations can lead to overfitting, so the same data will generate fewer gradients. This is disadvantageous for GS gradient attacks. For hidden size, a larger hidden size will retain more gradients, which is conducive to GS attacks and can lead to more privacy leaks.

The commonly used activation functions in regression problems are Sigmoid, Relu, Leaky Relu, and Tanh [129]. It is required that the derivatives of the activation function are non-zero. The derivatives are shown as follows.

$$\begin{aligned}(\text{Sigmoid})' &= (\frac{1}{1+\text{e}^{-i}})' \\ &= \frac{1'(1+\text{e}^{-i}) - 1(1+\text{e}^{-i})'}{(1+\text{e}^{-i})^2} \\ &= \frac{\text{e}^{-i}}{(1+\text{e}^{-i})^2},\end{aligned} \tag{5.19}$$

$$\begin{aligned}(\text{Tanh})' &= (\frac{\text{e}^{i}-\text{e}^{-i}}{\text{e}^{i}+\text{e}^{-i}})' \\ &= \frac{(\text{e}^{i}-\text{e}^{-i})'(\text{e}^{i}+\text{e}^{-i}) - (\text{e}^{i}-\text{e}^{-i})(\text{e}^{i}+\text{e}^{-i})'}{(\text{e}^{i}+\text{e}^{-i})^2} \\ &= \frac{(\text{e}^{i}+\text{e}^{-i})^2 - (\text{e}^{i}-\text{e}^{-i})^2}{(\text{e}^{i}+\text{e}^{-i})^2} \\ &= 1 - \frac{(\text{e}^{i}-\text{e}^{-i})^2}{(\text{e}^{i}+\text{e}^{-i})^2} \\ &= 1 - (\text{Tanh})^2\ ,\end{aligned} \tag{5.20}$$

$$(\text{LeakyRelu})' = (\max(0.01i, i))' = \begin{cases} 1\ (i < 0) \\ \text{None}\ (i = 0) \\ 0.01\ (i > 0)\ , \end{cases} \tag{5.21}$$

$$(\text{Relu})' = (\max(i, 0))' = \begin{cases} 0\ (i < 0) \\ \text{None}\ (i = 0) \\ 1\ (i > 0)\ , \end{cases} \tag{5.22}$$

where in practice, at i=0, the derivative of Relu and LeakyRelu is replaced by subgradient, which is a number in [0,1]. Generally, the derivative value is set as 0.

It is obvious that for Sigmoid, the derivatives cannot be zero. For Tanh, it is known that e^{i} and e^{-i} cannot be zero and e^{i} is a strictly increasing function, while e^{-i} is a strictly decreasing function. As i approaches positive infinity, e^{i} becomes much greater than e^{-i}, thus tanh(i) approaches 1. Similarly, as i approaches negative infinity, e^{-i} becomes much greater than e^{i}, leading tanh(i) to approach -1. Therefore, the range of tanh(i) and $(\tanh)^2$ is (-1,1) and the derivatives of Tanh can approach but will not equal 0 within its domain.

However, for Relu and Leaky Relu, there exists 0, so it will lead to zero gradients, therefore it is harder for the gradient optimizer to recover the original data.

5.4 Simulations setup

For software and hardware, PyTorch was used on a PC with a CPU Intel Core i7 11800H, GPU NVIDIA GeForce RTX 3080 and 32GB memory capacity.

5.4.1 Load and voltage prediction

For task I, load and voltage prediction, the individual household electric power consumption dataset is used in the simulation, including the Voltage and Load power data of four years [216]. The input size for the neural network is 6, which refers to the average load and voltage in the first 3 hours. The output predicts the average load and voltage for the next hour. The simulation is repeated four times

Table 5.1: Parameters for prediction system

Parameter in prediction system	**Values**
DNN input nodes	6
DNN hidden size	50 × 50
DNN output nodes	2
DNN learning rates	0.01-0.001
DNN training rounds	15000
DNN optimizer	Adam
DNN loss function	Mean-Squared-Error
DNN activation function	sigmoid/tanh/relu/leakyrelu

for activation functions to include sigmoid, tanh, relu and leakyrely at each time. The detailed parameters are shown in table 5.1.

5.4.2 Device action recommendation

For task II, device action recommendation, the UK-DALE (UK Domestic Appliance-Level Electricity) dataset is used, which provides high-precision power consumption data, including various types of household appliances such as refrigerators, washing machines, ovens, lighting, etc. [182]. The UK-DALE dataset was customized for the recommendation tasks in Chapter 3. For the neural network, the input size is 3, including device A, time, and device B. Each input is encoded into 1 × 32 embeddings and concatenated into a 1 × 96 embedding. The embedding vector is fed into 96 × 48 and 48 × 24 fully-connected layers. Finally, the output is a 0-1 number, which refers to the possibility of sequential use of electrical appliances A and B. The detailed parameter is shown in table 5.2.

5.4.3 Simulated GS attacks

Each task performs federated learning with other homes, and the gradients produced by the data during training are recorded. According to the gradient, GS attacks are performed with ADAM optimizer [211], and the federated learning and GS attack parameters are shown in table 5.3. Then, the recovered data will be paired according to the pairing principle in sub-section 5.2.2. The original data, recovered data and

Table 5.2: Parameters for the recommendation system

Parameter in recommendation system	**Values**
Device Embedding dimension	32
Relation Embedding dimension	32
Time Embedding dimension	32
Embedding input nodes	3
DNN input nodes	96
DNN hidden size	48×24
DNN output nodes	1
DNN learning rates	0.001
DNN training rounds	15000
DNN optimizer	Adam
DNN loss function	Mean-Squared-Error
DNN activation function	sigmoid and tanh

the Kendall coefficient between them need to be sent to the explanation module for an explanation.

Due to the wide distribution of data, further processing is required to ensure that the data are in an appropriate range. For instance, if the Voltage ranges from 230 V to 250 V, the data can be grouped into smaller ranges using a discretizer. For example, a voltage less than 235 V, between 235 V and 240 V, and greater than 240 V can be discretized into different groups, and 236 V will fall into the group between 235 V and 240 V. For the continuous case, the grouping method is applied for Voltage, Power and Embedding data, dividing the data intervals based on their maximum and minimum value. Subsequently, the *local_model* in line 18 of the **Algorithm 3** is used to analyze the importance of each feature with its weights, a simple Ridge Regression Model in the simulation. The final importance scores can be obtained by aggregating calculations for the dataset. For example, the summed weight for the feature Embedding <-0.07 in the *local_model* is 2.229 and 2.229 is also regarded as importance.

Although, in general, explainable machine learning methods are categorized into groups because it can only reflect general trends and have fluctuations in results, in the simulation of this chapter, for already discrete values (batch sizes, training iterations, hidden sizes), no grouping was performed, and each discrete value was

Table 5.3: Parameters for the federated learning and GS attack

Parameter in recommendation system	Values
Federated learning client number	4
learning local training rounds	1
GS optimizer	Adam
GS distance function	cosine similarity
GS loss function	Mean-Squared-Error
GS early stopping	5000 rounds
GS optimization iterations for each round	100
GS input size for prediction system	6
GS output size for prediction system	2
GS input size for recommendation system	96
GS output size for recommendation system	1

explained individually. Only for continuous values (load, voltage and embedding data), grouping was performed.

5.4.4 Explanation module

This sub-section mainly introduces the algorithm for the explanation module, including the first layer of LSTM and the second layer of explainable machine learning.

LSTM superparameter of layer I

The superparameter of layer I is shown in table 5.4, which is an LSTM network to fit the input size, training iteration, network hidden neuron size, original data and the Kendall coefficient between the original and recovered data.

Influence by DNN neuron size of layer I

In the prediction system, the number of neurons in the hidden layer of the DNN is progressively increasing from 10 to 150, indexed by 1 to 15. The detailed parameters are shown in table 5.5. A similar parameter is adopted in the recommendation system, and the number of neurons in the hidden layer of the DNN is also increasing from 6 to 312, indexed by 1 to 15. The detailed parameters are shown in table 5.6. Other than the hidden layer, for both applications, the other parameter is the same

Table 5.4: Simulation parameters for layer I

Parameter in LSTM and federated learning	**Values**
LSTM hidden nodes	50
LSTM learning rates	0.001
LSTM training rounds	0-15000
LSTM training batch size	1-30
LSTM optimizer	Adam
LSTM loss function	Mean-Squared-Error
LSTM state activation function	tanh
LSTM gate activation function	sigmoid

Table 5.5: Parameters for the hidden layer size of the prediction system

Index number	**Hidden layer size (number of neurons)**
1	10×10
2	20×20
3	30×30
4	40×40
5	50×50
6	60×60
7	70×70
8	80×80
9	90×90
10	100×100
11	110×110
12	120×120
13	130×130
14	140×140
15	150×150

as in table 5.1 and 5.2.

Meanwhile, as shown in table 5.6, the batch size ranges from 1 to 20, and the number of training iterations ranges from 0 to 10,000, numbered from 1 to 20, with every 500 training iterations forming one unit.

Superparameter of layer II

In the simulation, an explainable model is used to replace the non-explainable first layer LSTM in the second layer of the explanation model. The parameters of the

Table 5.6: Parameters for the hidden layer size of the recommendation system

Index number	**Hidden layer size (number of neurons)**
1	6 × 6
2	12 × 12
3	24 × 24
4	48 × 48
5	72 × 72
6	96 × 96
7	120 × 120
8	144 × 144
9	168 × 168
10	192 × 192
11	216 × 216
12	240 × 240
13	264 × 264
14	288 × 288
15	312 × 312

explainable model are as shown in table 5.7.

Table 5.7: Simulation parameters for layer II

Super parameters in explainable layer (layer 2)	**Values**
Federated learning batch size range	1-20
Federated learning training iterations range	0-10000
Number of features for training iterations	20
Federated learning hidden size	1-15
Distance metric	L2 distance
Model regressor	Ridge regression
Ridge regression alpha value	1
Ridge regression tolerance	0.001

The chosen batch size for federated learning ranges from 1 to 20, and the training iteration size ranges from 0 to 10,000. It is discretized into numbers from 0 to 20. For example, training iterations can be selected as 0, 500, and up to 10,000. As shown in table 5.5 and 5.6, the hidden size is quantified into numbers from 1 to 15. In the simulation, Ridge Regression, an explainable model, fits the non-explainable first-layer LSTM model. Table 5.7 also introduces the hyperparameters of the Ridge

Regression.

5.5 Results

This section presents the simulation results. Following the algorithmic process of SAFE-Home, the results of the GS attack on home applications are introduced. Subsequently, the results of the LSTM in the first layer of the explanation model were analyzed, determining the impact of different batch sizes, training iterations, activation functions, and hidden sizes on privacy leakage in a general view. Finally, the results of the second layer were analyzed, identifying key factors contributing to privacy leakage.

5.5.1 Result for GS attack on the applications

It is easy for the GS algorithm to compromise privacy in federated home applications. In the voltage and load forecasting system, Fig. 5.9 shows the recovery results for a continuous 24-hour period of a single household at the training iteration of 2000. During training, data is submitted to the server every 4 hours, resulting in a batch size of 4. There are two subplots in Fig. 5.9, with the first subplot representing load data and the second representing voltage data. It can be observed that the recovered data closely matches the original data, demonstrating that the privacy of users' daily data is highly susceptible to compromise. Even the fluctuations in peaking hours can be accurately recovered.

For the device action recommendation system, in Fig. 5.10, the input data are encoded as embeddings, and data from four appliances for each time period (1-28) were selected for recommendation and GS recovery. The simulation was conducted with each time period serving as a batch size. The yellow triangles represent the original data, the blue circles represent the recovered data, and the red 'x' marks are instances where the original and recovered data coincide. Names were labelled for the frequently recovered data. The simulation revealed that GS attacks can often recover the original appliance usage information through embedding, compromising user data privacy.

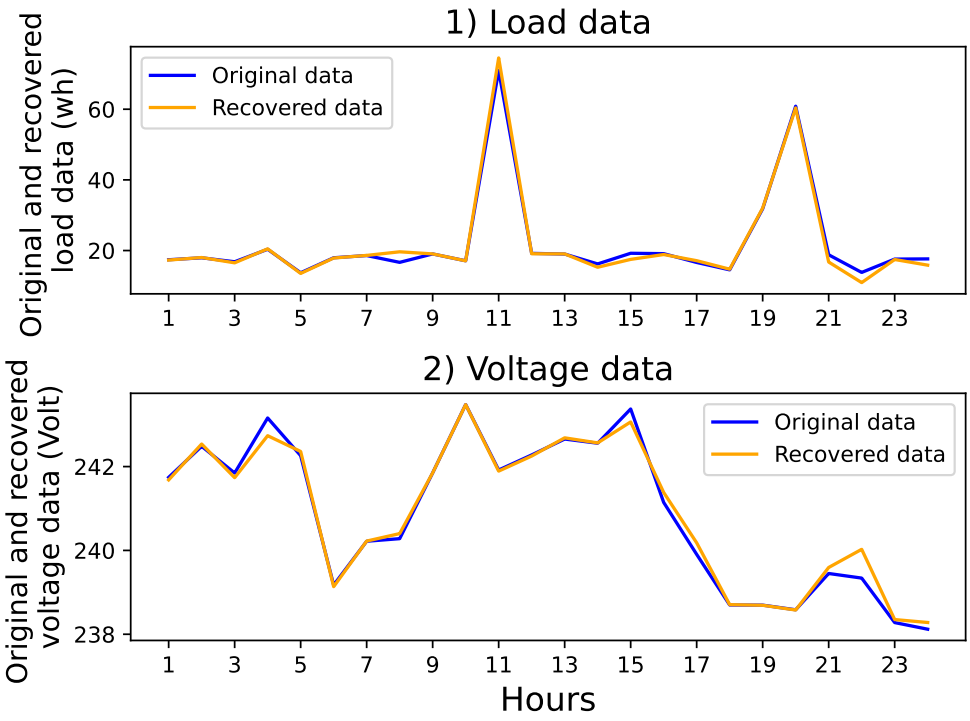

Figure 5.9: The original and recovered data for: 1) the load data and 2) the voltage data of a client. Simulation has found that the GS algorithm poses a significant threat to the prediction system and can almost completely restore the original data.

When the batch size or training iteration increases, hidden size decreases, or the activation may alleviate the privacy leakage, which will be discussed in the next subsection.

5.5.2 Result for layer I: The trend of the GS attack

Influence by batch sizes and training iterations

The well-fitted *network* in **Algorithm 3** was tested for evaluation. As shown in Fig. 5.11, the impact of batch size and training iteration on privacy leakage was represented in four 3D sub-graphs, where training iteration is between 1 to 30, referring to 1 × 500=500 rounds to 30 × 500=15000 rounds of training. The recovered data in the graph is divided into four colours: red (complete privacy leakage), orange-yellow (major privacy leakage), green (minor privacy leakage), and blue (almost no privacy leakage). It can be concluded that as the batch size and training iteration increase, privacy leakage gradually decreases. The most severe

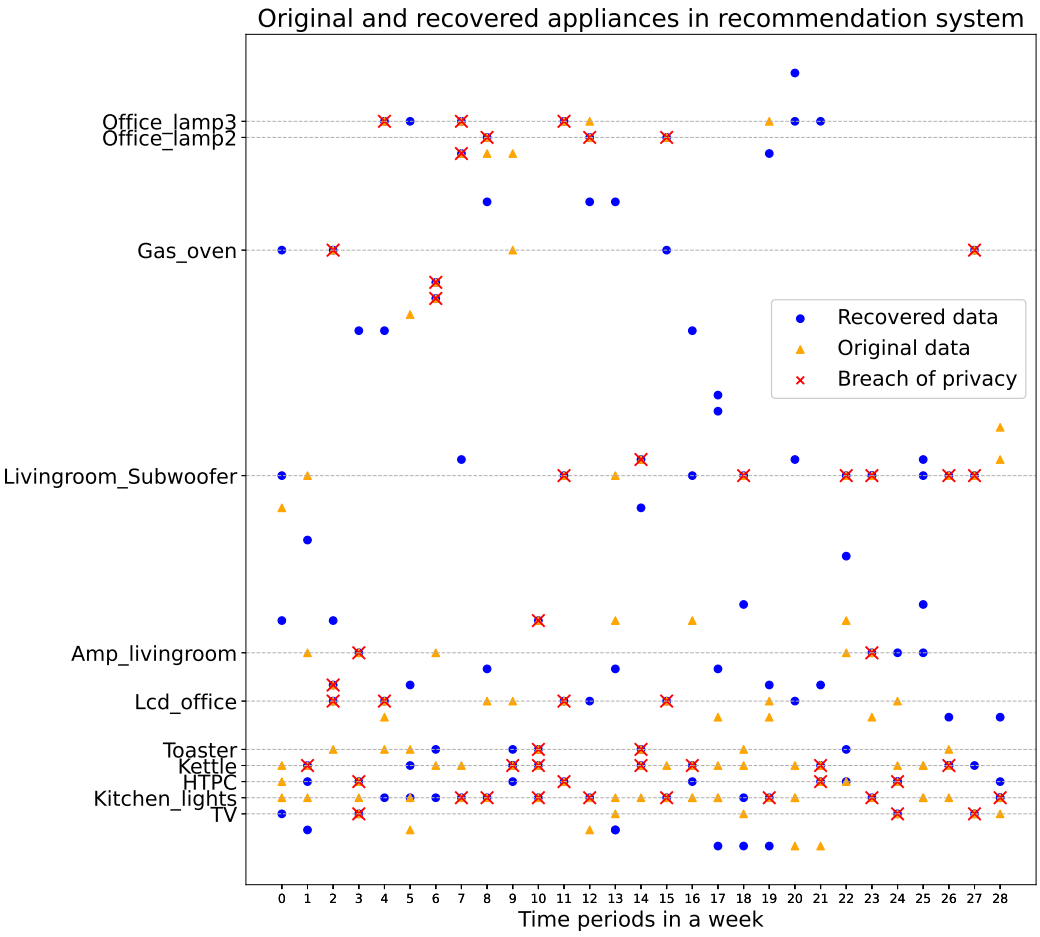

Figure 5.10: The original and recovered devices data for the recommendation system. It can be seen that the usage information of some appliances can be recovered. The GS attack poses a possible threat to recommendation systems. The graph labels appliances that have been recovered more than twice.

leakage situations for prediction systems occur when the batch size and training iteration are relatively small.

As shown in Fig. 5.12, after testing the recommendation system trained in **Algorithm 3**, it can be found that with different datasets, its trend is the same as that of Fig. 5.11, this demonstrates the generalization capability of the SAFE-Home algorithm. The result of Fig. 5.12 has 795 blue points, much more than the prediction system (blue points between 0 and 8). This is because the input is encoded into embeddings with higher dimensions, making the training data harder for the GS algorithm to retrieve.

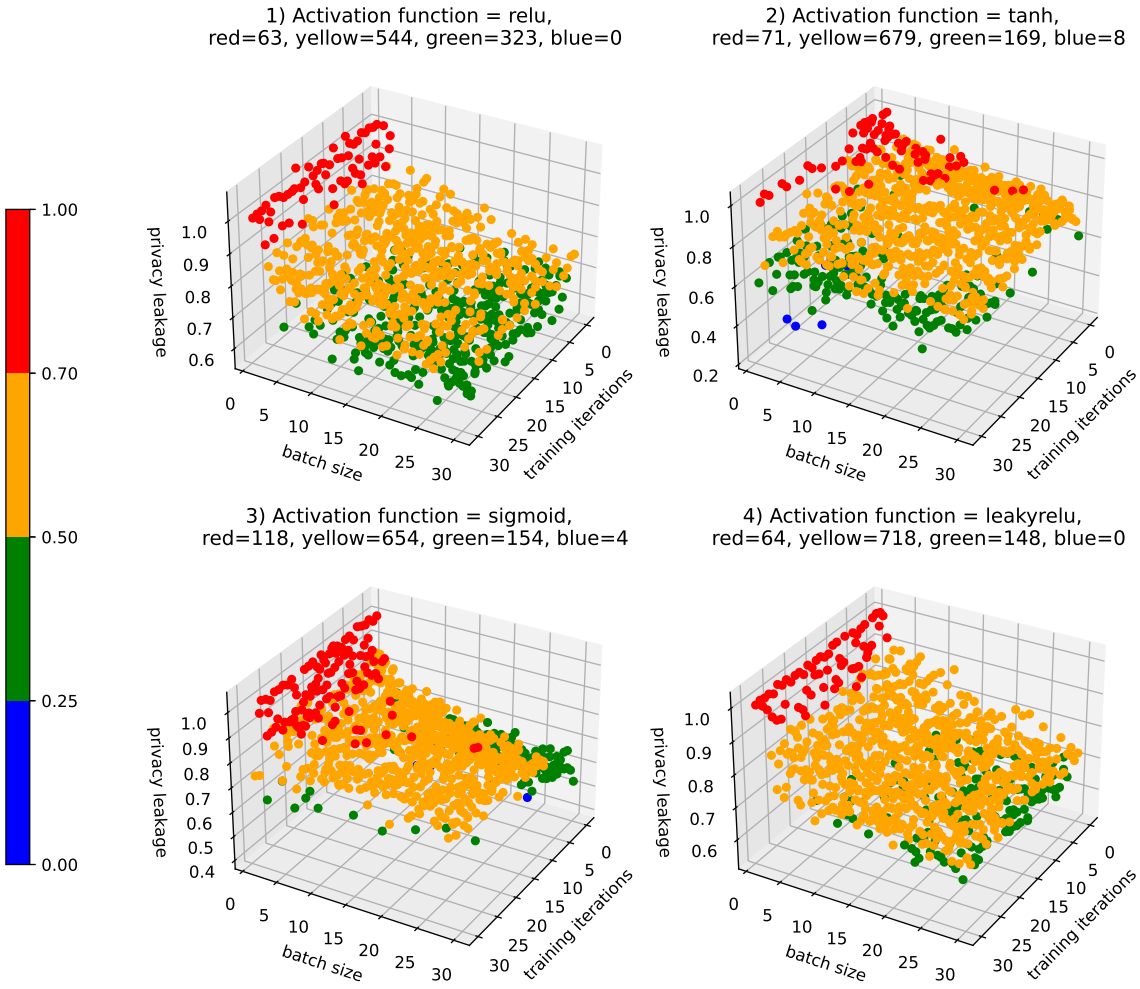

Figure 5.11: The explanation results of the GS attack on the prediction system with activation functions 1. Relu, 2. Tanh, 3. Sigmoid and 4. LeakyRelu (the Red colour denotes complete privacy leakage, orange signifies a major privacy leakage, green represents a minor privacy leakage, and blue indicates negligible privacy leakage).

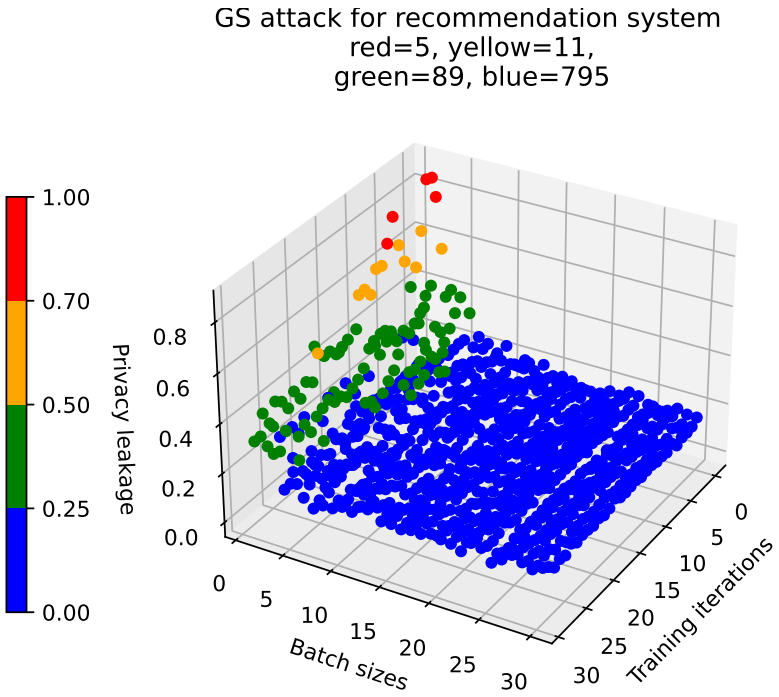

Figure 5.12: The recovery results of the GS attack on the recommendation system illustrates that only when the batch size and training iterations are relatively small does the recommendation system experience a more significant privacy leakage.

Influence by activation functions

The influence of activation functions is shown in Fig. 5.11. Different subgraphs represent the use of other activation functions. Based on data analysis, the activation functions 'Tanh' (red=71, yellow=679) and 'Sigmoid' (red=118, yellow=654) are the most prone to privacy leakage, with the most red and yellow points. However, after multiple simulation verifications, the data recovery is not very stable for sigmoid when training iterations=0. Please note that Sigmoid and Tanh share a common characteristic: they can accurately recover the original data even when the batch size is relatively large.

In contrast, Relu (red=63, yellow=544) and LeakyRelu (red=64, yellow=718) are more sensitive to batch size, and the original data cannot be recovered through attacks when the batch size is larger than 5. The activation function 'Relu' has the most minor leakage, with the most green points. This is because 'Relu' removes negative values, thus cutting off some of the gradients that GS attacks require to

use for gradient inversion. More gradient information will assist the GS algorithm in restoring data from the back-propagation of the model [129]. This simulation result/finding is consistent with the theoretical analysis presented in section 5.3.

Influence by DNN neuron size

From Figure 5.13 and 5.14, it is noteworthy that the trend of privacy leakage does not significantly increase or decrease with the change in system size in the simulations. Figure 5.13 illustrates the degree of privacy leakage in a recommendation system as the system size varies, where the trend of the privacy leakage cannot be distinguished clearly. Notably, the right subfigure shows more privacy leakage when the system size is around 12. Similarly, Figure 5.14 depicts the degree of privacy leakage in a prediction system as the system size varies, the right subfigure indicates significant privacy leakage when the system size is relatively large.

Other than that, when examining the impact of DNN hidden layer sizes on privacy leakage, what Fig. 5.13 and 5.14 have demonstrated is only one facet of the overall results, i.e., when either the results for training iteration equal 1000 or batch size equals 2. Therefore, further results are attached in the appendix. Since the trend changes are not obvious and cannot be easily distinguished, conclusions will require further in-depth exploration at the second level of the explainable model in the next layer of the model.

5.5.3 Result for layer II: Explainable analysis

In the original test, an analysis using explainable machine learning was conducted for cases where the recovered data and the original data are extremely similar (the Kendall coefficient between them is approximately equal to 1). After summarizing all the analyses, the weights are calculated to derive Fig. 5.15 and 5.16. Including:

- Batch Size: In both graphs, the impact of privacy leakage decreases as the batch size increases. The greatest privacy leakage occurs when the batch size is less than 3. Additionally, the variance for each point is relatively small.

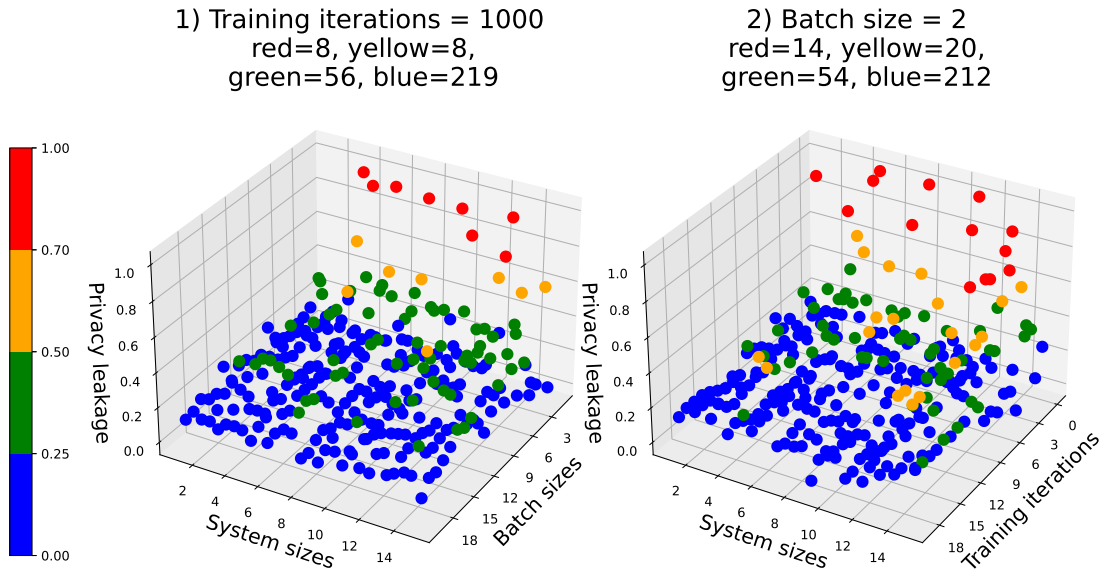

Figure 5.13: The influence of different system structures in the household recommendation system. 1) System size and batch size trends when the number of training iterations are 1000. 2) System size and number of training iterations trends when the batch size is 2. It can be seen from the graph that as the network structure increases, the trend of privacy leakage are hard to distinguish.

- Training Iterations: Similar to batch size, fewer training iterations are conducive to privacy leakage. The trend expressed in Fig. 5.15 and 5.16 are similar. The simulation found that the variance for training iterations is slightly larger than for batch size, especially in recommendation systems.

- Hidden Size: From both graphs, it can be inferred that as the hidden size of the neural network increases, the possibility of privacy leakage gradually increases (Although there are fluctuations). The variance in recommendation systems is larger compared to prediction systems.

- Voltage, Loads, and Embeddings Values: Based on three graphs, it can be deduced that extreme values (either too large or too small) in voltage, loads, and embeddings are prone to privacy leakage. For example, privacy leakage is more likely when voltage values are greater than 243 volts, load values are greater than 110 wh or less than 20 wh, or embedding values are greater

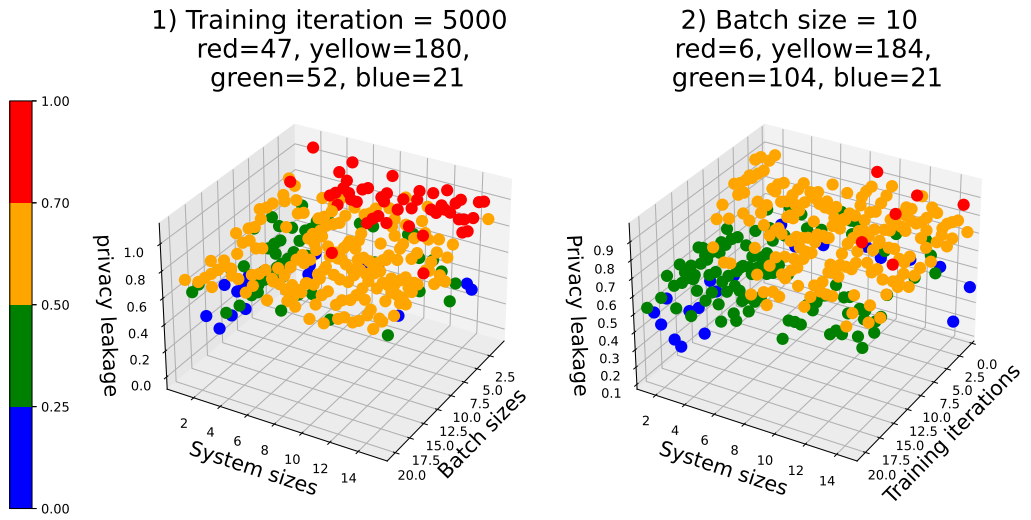

Figure 5.14: The influence of different system structures in the prediction system. 1) The trend of system size and batch size changes when the number of training iterations are 5000. 2) The changes in system size and in the number of training iterations when the batch size is 10.

than 0.6 or less than -0.2. Essentially, more distinctive or extreme values are more susceptible to privacy leakage. Additionally, the magnitude of values introduces greater uncertainty than other factors in privacy leakage. This aspect is evident in the subplot for embeddings in the recommendation system, which shows that embeddings have the largest variance.

The simulation also found greater uncertainty when considering the impact of privacy leakage on recommendation systems. This is because the use of embeddings leads to higher dimensions and more difficulty in data recovery. For prediction systems, the most critical factors affecting privacy are training iterations and hidden size. In contrast, for recommendation systems, the most crucial factor impacting privacy is batch size because the input dimension grows rapidly as the batch size grows.

Please note that in Fig. 5.15 and Fig. 5.16, due to the complexity of the deep learning model and the limitations of LIME algorithm locality, the results derived

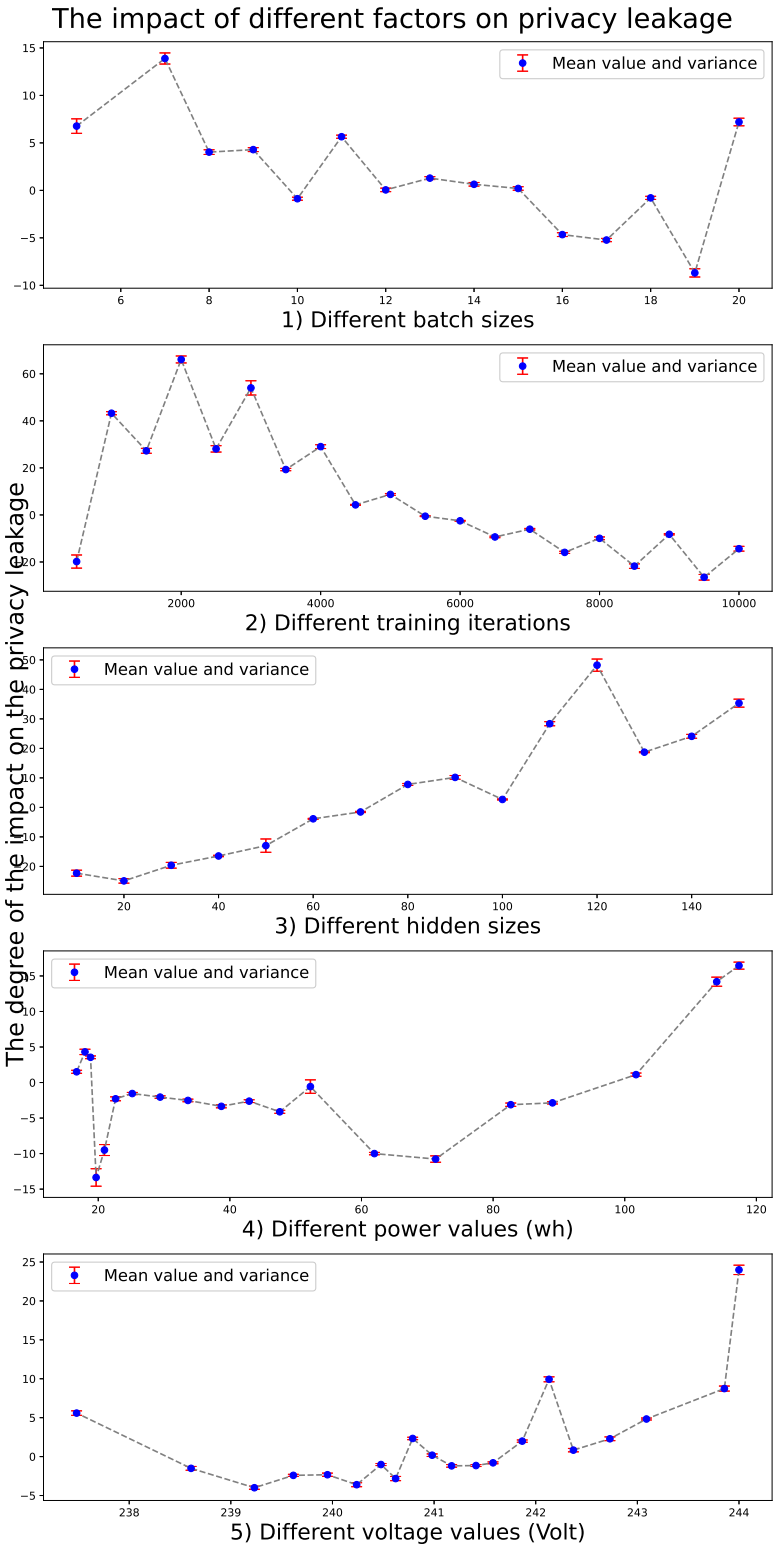

Figure 5.15: The recovery results of the prediction system. Including 1. different batch sizes, 2. different training iterations, 3. different hidden sizes, 4. different power values and 5. different voltage values. It can be observed that smaller training iterations, larger hidden sizes, and smaller batch sizes can lead to privacy leakage. Additionally, relatively large or small values of voltage and load data are easier to cause privacy leakage.

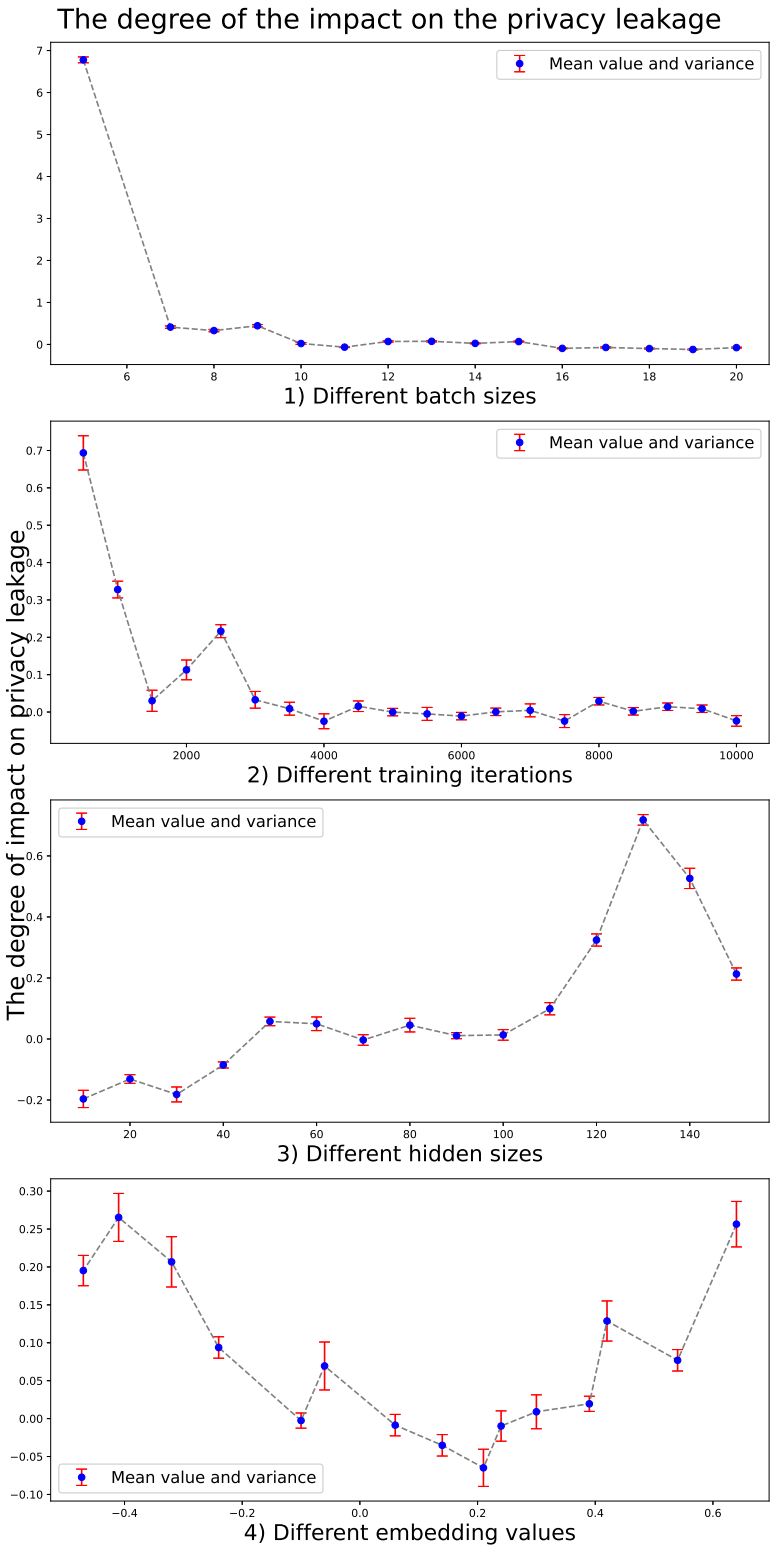

Figure 5.16: The explanation results of the recommendation system. Including 1. different batch sizes, 2. different training iterations, 3. different hidden sizes, and 4. different embedding values. It has been found that a smaller batch size, smaller training iterations, and a larger hidden size can lead to privacy leakage. Additionally, a larger or smaller embedding value also contributes to privacy leakage.

from layer 2 exhibit fluctuations, which can only reflect the general trend rather than the precise trend. To eliminate or mitigate fluctuations, the data can be further grouped, and the primary factors influencing privacy leakage can be filtered and focused. A method to alleviate fluctuation could be to divide the range into four groups: 0-40, 40-80, 80-120, and 120-150. However, these grouped forms of expressions will lose some information; consequently, they are not adopted in this simulation.

The results are consistent with the conclusions in the previous section 5.3, indicating that either a smaller batch size and training iteration or a larger hidden size will positively affect data recovery (privacy leakage) under the GS attack.

5.6 Conclusions

Building upon the home federated learning applications in the previous two chapters, this chapter further explores the extent and likelihood of privacy leakage through the proposed framework, SAFE-Home. In SAFE-Home, federated learning privacy measurement methods are proposed based on the GS algorithms, explainable machine learning algorithms, and various privacy metrics.

From the simulations, it can be concluded that under the voltage and load forecasting problem, as well as the device action recommendation problem, the privacy of federated learning can be easily compromised by the GS algorithm. The super parameters and activation functions of federated learning influence the performance of gradient inversion algorithms.

The contributions of this Chapter are as follows:

- The GS algorithm on household device action recommendations system as well as the load and voltage forecasting is carried out. These home AI scenarios are derived from Chapters 3 and 4. Simulation results indicate that even when using federated learning, there is still a risk of privacy leakage, posing a threat to the privacy concerns of household data.

- Leveraging the findings from implementing the GS algorithm and integrating explainable machine learning, a new algorithm in SAFE-Home is proposed to

predict the privacy leakage and explain the factors for potential privacy fears under various conditions when encountering the gradient inversion algorithm.

- Strategic advice for privacy preservation is derived. The simulations revealed that a smaller batch size and training iterations or a larger hidden neuron size, along with non-zero derivatives of the activation functions, are more likely to lead to privacy leakage. Extreme values are more easily recoverable.

Recommendations to protect residential privacy:

- Batch size: Synchronize the local model to the server with a larger batch size, because the simulation found that a smaller batch size contributes to the higher possibility of privacy leakage.

- Training iterations: The simulation found that fewer training iterations contribute to privacy leakage, so it is recommended to avoid providing the training data at the beginning of the local training or the training after mixing with other users.

- Activation functions: Try to use the activation function that has the zero or near to zero derivative, like Relu or LeakyRelu. Sigmoid or tanh will have the largest privacy leakage.

- Size of the neural network: Larger neural network architectures often contain more gradient information, which can lead to privacy leakage. Therefore, it is essential to choose the size of the neural network structure judiciously.

- Input dimension: Try to choose a larger input dimension, like using high-dimension embedding.

To conclude, simulations reveal that federated learning cannot entirely safeguard user data privacy. Defences against potential attacks are needed to minimize the likelihood of data leakage. Additionally, as various attack algorithms emerge, it is crucial to remain vigilant and devise corresponding strategies for potential privacy leaks in the future.

CHAPTER 6

Conclusions and future directions

6.1 Conclusions

This thesis proposes smart community algorithms for recommendations in smart homes and the scheduling in home microgrids while considering privacy protection.

As described in Chapter 1, the overall structure is divided into three parts. The first part is the smart home part, which is responsible for household device action recommendations and energy optimization according to the recommendation lists. The second part is the home microgrid part, developing a scheduling algorithm for optimizing the electricity purchase requirements in a home microgrid. The final part is dedicated to privacy protection, enhancing the overall privacy of the system. Chapter 2 described the related background and literature review.

In Chapter 3, a household device action recommendation framework, DARK, is established. The resident's daily appliance usage data is organized as a knowledge graph through data processing. The improved KGAT algorithm is used to train embeddings, aggregate graph information, and make recommendations. Unlike other household device action recommendation systems, this system considers the recommendations' interpretability and demand response optimization. The ef-

fectiveness of the proposed algorithm is validated by comparing it with algorithms based on DNN, RNN, CNN, and traditional KGAT through the use of simulations. It has been shown that with the DARK framework, the averaged recommendation accuracy achieved is 93.4%, and the optimized power can be reduced by up to 20%.

Chapter 4 establishes an energy scheduling system for the home microgrid. The system predicts the next-hour's environment, including the electricity prices, carbon emissions, and load consumption in the next hour. Based on this environment, a DQN algorithm is proposed for optimum electricity purchase decisions to optimize the accumulated electricity cost and carbon emissions. Unlike other DRL-based optimizations, the proposed algorithm connects multiple DQN models through federated learning, fully utilizing the information of each client's DQN while protecting privacy. Additionally, it provides each client with multiple optimization direction choices, making the model adjustable. The proposed algorithm is compared with the standalone DQN and multi-agent DQN models. It is found that federated DQN achieves a similar level of optimization as multi-agent DQN. Compared with a standalone DQN, the carbon emission decreased by 8.2%, and the electricity costs reduced by 10.7%.

In Chapter 5, a framework called SAFE-Home, is established to identify the conditions for privacy leakage. The testbed is based on federated learning in the household device action recommendation system and on the microgrid forecasting system. The first layer of SAFE-Home is based on Long Short-Term Memory, predicting the extent of possible privacy leakage under different conditions. The second layer is based on explainable machine learning with a customized loss function, which further pinpoints the potential circumstances leading to privacy leakage. The proposed algorithm is validated through tests using the UK-DALE dataset for home appliance recommendation systems and a household load dataset for forecasting residential loads and voltage, as well as this chapter's theoretical analysis.

In this thesis, the proposed AI smart community systems are closely related to residents' daily lives. Therefore, data security and privacy protection cannot be overlooked. While proposing algorithms, special attention has been paid to privacy protection concerns and the interpretability (or explainability) of the algorithms,

thus enhancing the practicality and usability of the algorithms.

6.2 Future research directions

For the home appliance recommendation system in Chapter 3, future research directions include:

- Utilizing data from a broader range of sources to build more complex graphs and maximize the effectiveness of knowledge graph algorithms.
- Developing more sophisticated demand response algorithms for modelling home heating systems. In the current recommendation system, appliances like refrigerators and water heaters are not manually controlled, so they have yet to be applied in the proposed algorithm.
- For long-tail data, such as home appliances that are used less frequently, incorporating human knowledge to improve recommendation accuracy needs further exploration.
- Trialing the proposed algorithm in more households and real-world environments, and set up questionnaires to test the correctness of the model's interpretation algorithm.
- Research on how to make recommendations in the absence of sub-electricity meters. Non-intrusive load monitoring algorithms could be used, or other recommendation methods could be employed.

For the home microgrid scheduling system in Chapter 4, the following future research directions are suggested:

- Energy trading between multiple households is an interesting topic for research, especially how to trade automatically. Mechanisms such as blockchain could be used to collaborate to realize a smart trading system.
- Since DQN relies on load prediction, more accurate predictions on renewable energy, electricity prices, and carbon emissions will enhance its performance, which is a possible research topic.

- A mechanism could be designed to make the bi-objective optimization's adjustable mechanism more detailed, for example, by creating user profiles based on actual usage patterns.

For the privacy protection system, SAFE-Home, proposed in Chapter 5, the following future research directions are suggested:

- The proposed framework could be applicable in simulating other types of federated learning attacks, such as membership inference attacks, GAN-based attacks, etc. Various attacks can be simulated for comparison, providing a more comprehensive privacy protection overview.

- The framework could be tested in other home-based federated learning scenarios, and the applicability of the proposed algorithm could be assessed in various scenarios.

- Further protection of privacy is a concern, and various privacy-preserving algorithms, such as differential privacy and encryption, could be employed to make privacy protection more comprehensive.

Bibliography

[1] Statista, "Statist data of smart home, 2024-2028." [Online] https://www.statista.com/outlook/dmo/smart-home/worldwide. Accessed August 01, 2024.

[2] A. Chakraborty, M. Islam, F. Shahriyar, S. Islam, H. U. Zaman, and M. Hasan, "Smart home system: a comprehensive review," *Journal of Electrical and Computer Engineering*, vol. 2023, no. 1, p. 7616683, 2023.

[3] the European Union Emission, "Statist data of carbon emissions, 2024-2028." [Online] https://climate.ec.europa.eu/eu-action/eu-emissions-trading-system-eu-ets/what-eu-ets_en#our-climate-ambition-for-2030. Accessed August 01, 2024.

[4] G. Dileep, "A survey on smart grid technologies and applications," *Renewable energy*, vol. 146, no. 1, pp. 2589–2625, 2020.

[5] M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, "A review of smart home applications based on internet of things," *Journal of network and computer applications*, vol. 97, no. 1, pp. 48–65, 2017.

[6] G. D. P. Regulation, "General data protection regulation (gdpr)," *Intersoft Consulting, Accessed in October*, vol. 24, no. 1, p. 1, 2018.

[7] I. G. P. Team, *EU general data protection regulation (gdpr)–an implementation and compliance guide*, vol. 1. IT Governance Ltd, 2020.

[8] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 1, p. 1, 2021.

[9] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, no. 1, p. 1, 2019.

[10] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[11] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of cleaner production*, vol. 140, no. 1, pp. 1454–1464, 2017.

[12] B. K. Sovacool and D. D. F. Del Rio, "Smart home technologies in europe: A critical review of concepts, benefits, risks and policies," *Renewable and sustainable energy reviews*, vol. 120, no. 1, p. 109663, 2020.

[13] G. Shahgholian, "A brief review on microgrids: Operation, applications, modeling, and control," *International Transactions on Electrical Energy Systems*, vol. 31, no. 6, p. e12885, 2021.

[14] K. Taghizad-Tavana, M. Ghanbari-Ghalehjoughi, N. Razzaghi-Asl, S. Nojavan, and A. a. Alizadeh, "An overview of the architecture of home energy management system as microgrids, automation systems, communication protocols, security, and cyber challenges," *Sustainability*, vol. 14, no. 23, p. 15938, 2022.

[15] T. Malche and P. Maheshwary, "Internet of things (IoT) for building smart home system," in *Proceedings of the 2017 International conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC)*, pp. 65–70, Palladam,Tirupur District, Tamilnadu, India, IEEE, 2017.

[16] S. Das, S. Namasudra, S. Deb, P. M. Ger, and R. G. Crespo, "Securing iot-based smart healthcare systems by using advanced lightweight privacy-preserving authentication scheme," *IEEE Internet of Things Journal*, vol. 1, no. 1-15, pp. 1–1, 2023.

[17] K. Rasch, "An unsupervised recommender system for smart homes," *Journal of Ambient Intelligence and Smart Environments*, vol. 6, no. 1, pp. 21–37, 2014.

[18] H. Chen, X. Xie, W. Shu, and N. Xiong, "An efficient recommendation filter model on smart home big data analytics for enhanced living environments," *Sensors*, vol. 16, no. 10, p. 1706, 2016.

[19] N. Belghini, N. Gouttaya, W. B. Bennani, and A. Sayouti, "Pervasive recommender system for smart home environment," *International Journal of Applied Information Systems*, vol. 10, no. 9, pp. 1–7, 2016.

[20] J. Reyes-Campos, G. Alor-Hernández, I. Machorro-Cano, J. O. Olmedo-Aguirre, J. L. Sánchez-Cervantes, and L. Rodríguez-Mazahua, "Discovery of resident behavior patterns using machine learning techniques and iot paradigm," *Mathematics*, vol. 9, no. 3, p. 219, 2021.

[21] H. Jeon, J. Kim, H. Yoon, J. Lee, and U. Kang, "Accurate action recommendation for smart home via two-level encoders and commonsense knowledge,"

in *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, vol. 1, pp. 832–841, Atlanta GA USA, 2022.

[22] K. Osman, M. Štefić, T. Alajbeg, and M. Perić, "Approach in development and implementation of automatic control system in a smart building," *Energy and Buildings*, vol. 1, no. 1, p. 113200, 2023.

[23] I. Varlamis, C. Sardianos, C. Chronis, G. Dimitrakopoulos, Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, "Smart fusion of sensor data and human feedback for personalized energy-saving recommendations," *Applied Energy*, vol. 305, no. 1, p. 117775, 2022.

[24] Y. Yao, M. M. Kamani, Z. Cheng, L. Chen, C. Joe-Wong, and T. Liu, "FedRule: Federated rule recommendation system with graph neural networks," *arXiv preprint arXiv:2211.06812*, vol. 6, no. 1, pp. 1–14, 2023.

[25] S. M. Ali, J. C. Augusto, D. Windridge, and E. Ward, "A user-guided personalization methodology to facilitate new smart home occupancy," *Universal Access in the Information Society*, vol. 22, no. 3, pp. 869–891, 2023.

[26] T. Dilekh, S. Benharzallah, A. Mokeddem, S. Kerdoudi, *et al.*, "Dynamic context-aware recommender system for home automation through synergistic unsupervised and supervised learning algorithms," *Acta Informatica Pragensia*, vol. 1, no. 1, p. 1, 2024.

[27] R. F. Atallah, C. M. Assi, and M. J. Khabbaz, "Scheduling the operation of a connected vehicular network using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 5, pp. 1669–1682, 2018.

[28] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE transactions on smart grid*, vol. 10, no. 4, pp. 3698–3708, 2018.

[29] B. Wang, Y. Li, W. Ming, and S. Wang, "Deep reinforcement learning method for demand response management of interruptible load," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3146–3155, 2020.

[30] H.-M. Chung, S. Maharjan, Y. Zhang, and F. Eliassen, "Distributed deep reinforcement learning for intelligent load scheduling in residential smart grids," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2752–2763, 2020.

[31] Y. Liu, D. Zhang, and H. B. Gooi, "Optimization strategy based on deep reinforcement learning for home energy management," *CSEE Journal of Power and Energy Systems*, vol. 6, no. 3, pp. 572–582, 2020.

[32] Y. Ye, D. Qiu, X. Wu, G. Strbac, and J. Ward, "Model-free real-time autonomous control for a residential multi-energy system using deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3068–3082, 2020.

[33] S. Bahrami, Y. C. Chen, and V. W. Wong, "Deep reinforcement learning for demand response in distribution networks," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1496–1506, 2020.

[34] T. Yang, L. Zhao, W. Li, and A. Y. Zomaya, "Dynamic energy dispatch strategy for integrated energy system based on improved deep reinforcement learning," *Energy*, vol. 235, no. 1, p. 121377, 2021.

[35] T. A. Nakabi and P. Toivanen, "Deep reinforcement learning for energy management in a microgrid with flexible demand," *Sustainable Energy, Grids and Networks*, vol. 25, no. 1, p. 100413, 2021.

[36] L. Zhao, T. Yang, W. Li, and A. Y. Zomaya, "Deep reinforcement learning-based joint load scheduling for household multi-energy system," *Applied Energy*, vol. 324, no. 1, p. 119346, 2022.

[37] J. Wang, C. Guo, C. Yu, and Y. Liang, "Virtual power plant containing electric vehicles scheduling strategies based on deep reinforcement learning," *Electric power systems research*, vol. 205, no. 1, p. 107714, 2022.

[38] M. Ren, X. Liu, Z. Yang, J. Zhang, Y. Guo, and Y. Jia, "A novel forecasting based scheduling method for household energy management system based on deep reinforcement learning," *Sustainable Cities and Society*, vol. 76, no. 1, p. 103207, 2022.

[39] K. Wang, H. Wang, J. Yang, J. Feng, Y. Li, S. Zhang, and M. O. Okoye, "Electric vehicle clusters scheduling strategy considering real-time electricity prices based on deep reinforcement learning," *Energy Reports*, vol. 8, no. 1, pp. 695–703, 2022.

[40] M. Alqahtani and M. Hu, "Dynamic energy scheduling and routing of multiple electric vehicles using deep reinforcement learning," *Energy*, vol. 244, no. 2, p. 122626, 2022.

[41] Y. Zhang, Q. Yang, D. An, D. Li, and Z. Wu, "Multistep multiagent reinforcement learning for optimal energy schedule strategy of charging stations in smart grid," *IEEE Transactions on Cybernetics*, vol. 1, no. 1, p. 1, 2022.

[42] Y. Lu, Y. Xiang, Y. Huang, B. Yu, L. Weng, and J. Liu, "Deep reinforcement learning based optimal scheduling of active distribution system considering distributed generation, energy storage and flexible load," *Energy*, vol. 271, no. 1, p. 127087, 2023.

[43] D. Liu, P. Zeng, S. Cui, and C. Song, "Deep reinforcement learning for charging scheduling of electric vehicles considering distribution network voltage stability," *Sensors*, vol. 23, no. 3, p. 1618, 2023.

[44] Y. Yang, T. Ma, H. Li, Y. Liu, C. Tang, and W. Pei, "Federated double dqn based multi-energy microgrid energy management strategy considering carbon emissions," *Global Energy Interconnection*, vol. 6, no. 6, pp. 689–699, 2023.

[45] N. G. Paterakis, O. Erdinç, and J. P. Catalão, "An overview of demand response: Key-elements and international experience," *Renewable and Sustainable Energy Reviews*, vol. 69, no. 1, pp. 871–891, 2017.

[46] D. Li, W.-Y. Chiu, H. Sun, and H. V. Poor, "Multiobjective optimization for demand side management program in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1482–1490, 2017.

[47] F. Pallonetto, M. De Rosa, F. Milano, and D. P. Finn, "Demand response algorithms for smart-grid ready residential buildings using machine learning models," *Applied energy*, vol. 239, no. 1, pp. 1265–1282, 2019.

[48] H. Li, Z. Wan, and H. He, "Real-time residential demand response," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4144–4154, 2020.

[49] F. Alfaverh, M. Denai, and Y. Sun, "Demand response strategy based on reinforcement learning and fuzzy reasoning for home energy management," *IEEE access*, vol. 8, no. 1, pp. 39310–39321, 2020.

[50] A. C. Duman, H. S. Erden, Ö. Gönül, and Ö. Güler, "A home energy management system with an integrated smart thermostat for demand response in smart grids," *Sustainable Cities and Society*, vol. 65, no. 1, p. 102639, 2021.

[51] Z. Chen, Y. Chen, R. He, J. Liu, M. Gao, and L. Zhang, "Multi-objective residential load scheduling approach for demand response in smart grid," *Sustainable Cities and Society*, vol. 76, no. 1, p. 103530, 2022.

[52] L. Wen, K. Zhou, W. Feng, and S. Yang, "Demand side management in smart grid: A dynamic-price-based demand response model," *IEEE Transactions on Engineering Management*, vol. 71, no. 1, pp. 1439–1451, 2024.

[53] H. J. Monfared, A. Ghasemi, A. Loni, and M. Marzband, "A hybrid price-based demand response program for the residential micro-grid," *Energy*, vol. 185, no. 6, pp. 274–285, 2019.

[54] M. Tostado-Véliz, P. Arévalo, S. Kamel, H. M. Zawbaa, and F. Jurado, "Home energy management system considering effective demand response strategies and uncertainties," *Energy Reports*, vol. 8, no. 1, pp. 5256–5271, 2022.

[55] D. Stanelyte, N. Radziukyniene, and V. Radziukynas, "Overview of demand-response services: A review," *Energies*, vol. 15, no. 5, p. 1659, 2022.

[56] S. Sharda, M. Singh, and K. Sharma, "A complete consumer behaviour learning model for real-time demand response implementation in smart grid," *Applied Intelligence*, vol. 52, no. 1, pp. 835–845, 2022.

[57] H. Shreenidhi and N. S. Ramaiah, "A two-stage deep convolutional model for demand response energy management system in iot-enabled smart grid," *Sustainable Energy, Grids and Networks*, vol. 30, no. 1, p. 100630, 2022.

[58] B. Yu, F. Sun, C. Chen, G. Fu, and L. Hu, "Power demand response in the context of smart home application," *Energy*, vol. 240, no. 1, p. 122774, 2022.

[59] M. J. Blaschke, "Dynamic pricing of electricity: Enabling demand response in domestic households," *Energy Policy*, vol. 164, no. 1, p. 112878, 2022.

[60] J. Huang, D. D. Koroteev, and M. Rynkovskaya, "Machine learning-based demand response in pv-based smart home considering energy management in digital twin," *Solar Energy*, vol. 252, no. 1, pp. 8–19, 2023.

[61] P. H. Mirzaee, M. Shojafar, H. Cruickshank, and R. Tafazolli, "Smart grid security and privacy: From conventional to machine learning issues (threats and countermeasures)," *IEEE access*, vol. 10, no. 1, pp. 52922–52954, 2022.

[62] M. B. Hossain, I. Natgunanathan, Y. Xiang, L.-X. Yang, and G. Huang, "Enhanced smart meter privacy protection using rechargeable batteries," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7079–7092, 2019.

[63] X. Jiajia, L. Liping, and B. Zhenmao, "Smart grid privacy protection using solar energy and rechargeable battery," *Energy Procedia*, vol. 158, no. 1, pp. 6170–6175, 2019.

[64] P. Bovornkeeratiroj, S. Iyengar, S. Lee, D. Irwin, and P. Shenoy, "Repel: A utility-preserving privacy system for iot-based energy meters," in *Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, vol. 1, pp. 79–91, IEEE, 2020.

[65] S. Desai, R. Alhadad, N. Chilamkurti, and A. Mahmood, "A survey of privacy preserving schemes in ioe enabled smart grid advanced metering infrastructure," *Cluster Computing*, vol. 22, no. 1, pp. 43–69, 2019.

[66] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, March 4-7, 2006.*, vol. 1, pp. 265–284, Springer, New York, NY, USA, 2006.

[67] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Differential privacy for power grid obfuscation," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1356–1366, 2019.

[68] M. U. Hassan, M. H. Rehmani, R. Kotagiri, J. Zhang, and J. Chen, "Differential privacy for renewable energy resources based smart metering," *Journal of Parallel and Distributed Computing*, vol. 131, no. 1, pp. 69–80, 2019.

[69] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multiparty computation," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, vol. 1, pp. 639–648, Philadelphia Pennsylvania USA, 1996.

[70] L. Sweeney, "k-anonymity: A model for protecting privacy," *International journal of uncertainty, fuzziness and knowledge-based systems*, vol. 10, no. 05, pp. 557–570, 2002.

[71] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, vol. 1, pp. 169–178, New York, NY, United States, 2009.

[72] W. Kong, J. Shen, P. Vijayakumar, Y. Cho, and V. Chang, "A practical group blind signature scheme for privacy protection in smart grid," *Journal of Parallel and Distributed Computing*, vol. 136, no. 1, pp. 29–39, 2020.

[73] M. Baza, M. Pazos-Revilla, A. Sherif, M. Nabil, A. J. Aljohani, M. Mahmoud, and W. Alasmary, "Privacy-preserving and collusion-resistant charging coordination schemes for smart grids," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2226–2243, 2021.

[74] Z. Abreu and L. Pereira, "Privacy protection in smart meters using homomorphic encryption: An overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 12, no. 4, p. e1469, 2022.

[75] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artifical intelligence and statistics*, vol. 1, no. 1, pp. 1273–1282, 2017.

[76] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *arXiv*, vol. 2, no. 1, pp. 429–450, 2020.

[77] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proceedings of the International conference on learning representation*, pp. 1–13, Addis Ababa, Ethiopia, 2020.

[78] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proceedings of ICLR 2020 conference*, pp. 1–12, Virtual Conference, 2020.

[79] Y. Liu, T. Fan, T. Chen, Q. Xu, and Q. Yang, "Fate: An industrial grade platform for collaborative learning with data protection," *Journal of Machine Learning Research*, vol. 22, no. 226, pp. 1–6, 2021.

[80] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.

[81] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12806–12825, 2021.

[82] X. Zhang, F. Fang, and J. Wang, "Probabilistic solar irradiation forecasting based on variational bayesian inference with secure federated learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7849–7859, 2020.

[83] N. Gholizadeh and P. Musilek, "Federated learning with hyperparameter-based clustering for electrical load forecasting," *Internet of Things*, vol. 17, no. 1, p. 100470, 2022.

[84] O. Bouachir, M. Aloqaily, Ö. Özkasap, and F. Ali, "Federatedgrids: Federated learning and blockchain-assisted p2p energy sharing," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 424–436, 2022.

[85] S. Otoum, I. Al Ridhawi, and H. Mouftah, "A federated learning and blockchain-enabled sustainable energy trade at the edge: A framework for industry 4.0," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3018–3026, 2022.

[86] Z. Wang, P. Yu, and H. Zhang, "Privacy-preserving regulation capacity evaluation for hvac systems in heterogeneous buildings based on federated learning and transfer learning," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, p. 1, 2022.

[87] J. Lin, J. Ma, and J. Zhu, "Hierarchical federated learning for power transformer fault diagnosis," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, no. 1, pp. 1–11, 2022.

[88] Y. Li, X. Wei, Y. Li, Z. Dong, and M. Shahidehpour, "Detection of false data injection attacks in smart grid: A secure federated deep learning approach," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4862–4872, 2022.

[89] Y. Zhang, G. Tang, Q. Huang, Y. Wang, K. Wu, K. Yu, and X. Shao, "Fednilm: Applying federated learning to nilm applications at the edge," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, p. 1, 2022.

[90] W.-T. Lin, G. Chen, and Y. Huang, "Incentive edge-based federated learning for false data injection attack detection on power grid state estimation: A novel mechanism design approach," *Applied Energy*, vol. 314, no. 1, p. 118828, 2022.

[91] V. Venkataramanan, S. Kaza, and A. M. Annaswamy, "Der forecast using privacy-preserving federated learning," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2046–2055, 2022.

[92] M. N. Fekri, K. Grolinger, and S. Mir, "Distributed load forecasting using smart meter data: Federated learning with recurrent neural networks," *International Journal of Electrical Power and Energy Systems*, vol. 137, no. 1, p. 107669, 2022.

[93] H. Moayyed, A. Moradzadeh, B. Mohammadi-Ivatloo, A. P. Aguiar, and R. Ghorbani, "A cyber-secure generalized supermodel for wind power forecasting based on deep federated learning and image processing," *Energy Conversion and Management*, vol. 267, no. 1, p. 115852, 2022.

[94] M. M. Badr, M. I. Ibrahem, M. Mahmoud, W. Alasmary, M. M. Fouda, K. H. Almotairi, and Z. M. Fadlullah, "Privacy-preserving federated-learning-based net-energy forecasting," in *Proceedings of the SoutheastCon 2022*, vol. 1, pp. 133–139, Mobile, AL, USA, 2022.

[95] Z. Duan, Y. Qiao, S. Chen, X. Wang, G. Wu, and X. Wang, "Lightweight federated reinforcement learning for independent request scheduling in microgrids," in *Proceedings of the 2022 IEEE International Conference on Smart Internet of Things (SmartIoT)*, vol. 1, pp. 208–215, IEEE, Suzhou, China, 2022.

[96] Y. Du, N. Mendes, S. Rasouli, J. Mohammadi, and P. Moura, "Federated learning assisted distributed energy optimization," *arXiv preprint arXiv:2311.13785*, vol. 1, no. 1, p. 1, 2023.

[97] V. Veerasamy, L. M. I. Sampath, S. Singh, H. D. Nguyen, and H. B. Gooi, "Blockchain-based decentralized frequency control of microgrids using federated learning fractional-order recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, p. 1, 2023.

[98] P. Wiesner, R. Khalili, D. Grinwald, P. Agrawal, L. Thamsen, and O. Kao, "Fedzero: Leveraging renewable excess energy in federated learning," *arXiv preprint arXiv:2305.15092*, vol. 1, no. 1, p. 1, 2023.

[99] D. D. Sharma, "Asynchronous blockchain-based federated learning for tokenized smart power contract of heterogeneous networked microgrid system," *IET Blockchain*, vol. 1, no. 1, p. 1, 2023.

[100] I. Naidji, C. E. Choucha, and M. Ramdani, "Decentralized federated learning architecture for networked microgrids," vol. 1, no. 1, p. 1, 2023.

[101] V. Veerasamy, H. Qiu, A. M. Y. M. Ghias, K. Chauhan, H. D. Nguyen, and H. B. Gooi, "Federated-learning-based distributed frequency control against false data injection attack," *IEEE Transactions on Industrial Electronics*, vol. 1, no. 1, p. 1, 2023.

[102] A. X. R. Irudayaraj, N. I. A. Wahab, V. Veerasamy, M. Premkumar, V. K. Ramachandaramurthy, and H. B. Gooi, "Optimal frequency regulation in multi-microgrid systems using federated learning," in *Proceedings of the 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*, vol. 1, pp. 1–6, IEEE, Beijing, China, 2023.

[103] C. Ren, H. Yu, R. Yan, Q. Li, Y. Xu, D. Niyato, and Z. Y. Dong, "Secfedsa: A secure differential privacy-based federated learning approach for smart cyber-physical grid stability assessment," *IEEE Internet of Things Journal*, vol. 1, no. 1, p. 1, 2023.

[104] A. H. Bondok, M. Mahmoud, M. M. Badr, M. M. Fouda, M. Abdallah, and M. Alsabaan, "Novel evasion attacks against adversarial training defense for smart grid federated learning," *IEEE Access*, vol. 1, no. 1, p. 1, 2023.

[105] H. Moayyed, A. Moradzadeh, A. Mansour-Saatloo, B. Mohammadi-Ivatloo, M. Abapour, and Z. Vale, "A global cyber-resilient model for dynamic line rating forecasting based on deep federated learning," *IEEE Systems Journal*, vol. 1, no. 1, p. 1, 2023.

[106] Q. Xu and X. Shi, "FTL-EDGE: A feature federation transfer learning algorithm for single microgrid edge computing," in *Proceedings of the 2023 8th International Conference on Power and Renewable Energy (ICPRE)*, vol. 1, pp. 981–986, IEEE, Shanghai, China, 2023.

[107] Y. Li, S. He, Y. Li, Y. Shi, and Z. Zeng, "Federated multiagent deep reinforcement learning approach via physics-informed reward for multimicrogrid energy management," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 1, no. 1, p. 1, 2023.

[108] A. Zekiye and Ö. Özkasap, "The internet of energy systems: Blockchain and smart contracts meet federated learning," in *Proceedings of 2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, vol. 1, pp. 1–3, Dubai, United Arab Emirates, 2023.

[109] A. Zekiye and Ö. Özkasap, "Blockchain-based federated learning for decentralized energy management systems," in *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, vol. 1, pp. 186–193, Kuwait City, 2023.

[110] P. Whig, A. Velu, and P. Sharma, "Demystifying federated learning for blockchain: A case study," in *Demystifying Federated Learning for Blockchain and Industrial Internet of Things*, vol. 1, pp. 143–165, IGI Global, 2022.

[111] M. M. Badr, M. Mahmoud, Y. Fang, M. Abdulaal, A. J. Aljohani, W. Alasmary, and M. I. Ibrahem, "Privacy-preserving and communication-efficient energy prediction scheme based on federated learning for smart grids," *IEEE Internet of Things Journal*, vol. 1, no. 1, p. 1, 2023.

[112] M. Al-Quraan, A. Khan, A. Centeno, A. Zoha, M. A. Imran, and L. Mohjazi, "Fedtrees: A novel computation-communication efficient federated learning framework investigated in smart grids," *arXiv preprint arXiv:2210.00060*, vol. 1, no. 1, p. 1, 2022.

[113] A. Sundararajan, R. A. Bridges, M. Olama, and M. Ferrari, "A privacy-aware federated learning framework for distributed energy resource analytics in constrained environments," in *2023 IEEE PES Innovative Smart Grid Technologies Latin America (ISGT-LA)*, vol. 1, pp. 155–159, San Juan, Puerto Rico, 2023.

[114] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, no. 1, pp. 1–11, 2019.

[115] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," *Advances in Neural Information Processing Systems*, vol. 33, no. 1, pp. 16937–16947, 2020.

[116] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," *arXiv*, vol. 1, no. 1, pp. 1–5, 2020.

[117] W. Wei, L. Liu, M. Loper, K.-H. Chow, M. E. Gursoy, S. Truex, and Y. Wu, "A framework for evaluating gradient leakage attacks in federated learning," *arXiv preprint arXiv:2004.10397*, vol. 1, no. 1, pp. 1–5, 2020.

[118] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov, "See through gradients: Image batch recovery via gradinversion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 16337–16346, Nashville, TN, USA, 2021.

[119] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, vol. 1, pp. 1322–1333, Denver Colorado USA, 2015.

[120] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of the 2017 IEEE symposium on security and privacy (SP)*, pp. 3–18, San Jose, CA, USA, 2017.

[121] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, vol. 1, pp. 603–618, Dallas Texas USA, 2017.

[122] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[123] B. Mahesh, "Machine learning algorithms, a review," *International Journal of Science and Research (IJSR).*, vol. 9, no. 1, pp. 381–386, 2020.

[124] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, no. 1, pp. 1–10, 2017.

[125] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, no. 1, pp. 1877–1901, 2020.

[126] G. Bradski, "The opencv library.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[127] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[128] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, vol. 1, pp. 191–198, Boston, USA, 2016.

[129] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[130] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[131] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, vol. 1, no. 1, p. 1, 2014.

[132] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[133] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[134] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[135] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *Proceedings of the world wide web conference*, pp. 3307–3313, San Francisco, CA, USA, 2019.

[136] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[137] M. Volodymyr, K. Koray, S. David, A. A. Rusu, V. Joel, M. G. Bellemare, G. Alex, R. Martin, A. K. Fidjeland, and O. Georg, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–33, 2015.

[138] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, vol. 1, no. 1, pp. 1–10, 2016.

[139] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, no. 1, pp. 1–10, 2017.

[140] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, vol. 1, no. 1, pp. 1–10, 2017.

[141] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, no. 1, pp. 109–132, 2013.

[142] P. B. Thorat, R. M. Goudar, and S. Barve, "Survey on collaborative filtering, content-based filtering and hybrid recommendation system," *International Journal of Computer Applications*, vol. 110, no. 4, pp. 31–36, 2015.

[143] S. Rendle, "Factorization machines," in *Proceedings of 2010 IEEE International conference on data mining*, pp. 995–1000, IEEE, Sydney, Australia, 2010.

[144] Q. Zhang, J. Lu, and Y. Jin, "Artificial intelligence in recommender systems," *Complex and Intelligent Systems*, vol. 7, no. 1, pp. 439–457, 2021.

[145] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[146] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *Proceedings of 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, vol. 1, pp. 1–6, IEEE, Salerno, Italy, 2016.

[147] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, no. 1, pp. 1–14, 2013.

[148] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, vol. 1, pp. 701–710, New York USA, 2014.

[149] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, vol. 1, pp. 855–864, San Francisco California USA, 2016.

[150] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, Anchorage USA*, vol. 1, pp. 950–958, 2019.

[151] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of the Semantic Web: 15th International Conference, ESWC 2018, June 3–7, 2018, Proceedings 15*, vol. 1, pp. 593–607, Springer, Heraklion, Crete, Greece, 2018.

[152] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, and J. Trotta, "How to make latent factors interpretable by feeding factorization machines with knowledge graphs," in *Proceedings of the Semantic Web–ISWC 2019: 18th*

International Semantic Web Conference, October 26–30, 2019, Proceedings, Part I 18, pp. 38–56, Springer, Auckland, New Zealand, 2019.

[153] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv*, vol. 1, no. 1, p. 1, 2013.

[154] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, vol. 1, pp. 173–182, Perth Australia, 2017.

[155] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, vol. 1, no. 1, pp. 1–10, 2018.

[156] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, no. 1, pp. 237–285, 1996.

[157] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[158] S. E. Li, "Reinforcement learning for sequential decision and optimal control," *Springer*, vol. 1, no. 1, pp. 365–402, 2023.

[159] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings if the International conference on machine learning*, pp. 1995–2003, PMLR, New York City, NY, USA, 2016.

[160] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, Phoenix, Arizona, USA, 2016.

[161] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of International conference on machine learning*, vol. 1, pp. 1889–1897, PMLR, Lille, France, 2015.

[162] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, vol. 1, no. 1, p. 1, 2017.

[163] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, no. 1, p. 1, 2017.

[164] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, vol. 1, no. 1, p. 1, 2015.

[165] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," *IEEE Access*, vol. 8, no. 4, pp. 42200–42216, 2020.

[166] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you? explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, San Francisco California USA, 2016.

[167] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[168] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, vol. 1, pp. 618–626, Venice, Italy, 2017.

[169] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, no. 1, p. 1, 2017.

[170] P. R. Magesh, R. D. Myloth, and R. J. Tom, "An explainable machine learning model for early detection of parkinson's disease using lime on datscan imagery," *Computers in Biology and Medicine*, vol. 126, no. 1, p. 104041, 2020.

[171] J. Dieber and S. Kirrane, "Why model why? assessing the strengths and limitations of lime," *arXiv preprint arXiv:2012.00093*, vol. 1, no. 1, p. 1, 2020.

[172] Z. Zhou, G. Hooker, and F. Wang, "S-lime: Stabilized-lime for model explanation," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery and data mining*, vol. 1, pp. 2429–2438, Virtual Event, Singapore, 2021.

[173] A. Sangroya, M. Rastogi, C. Anantaram, and L. Vig, "Guided-lime: Structured sampling based hybrid approach towards explaining blackbox machine learning models.," in *Proceedings of CIKM (Workshops)*, vol. 1, p. 1, ONLINE, 2020.

[174] K. Lin and Y. Gao, "Model interpretability of financial fraud detection by group shap," *Expert Systems with Applications*, vol. 210, no. 1, p. 118354, 2022.

[175] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," *arXiv preprint arXiv:1711.05851*, vol. 1, no. 1, p. 1, 2017.

[176] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," *arXiv preprint arXiv:1707.06690*, vol. 1, no. 1, p. 1, 2017.

[177] Q. Ai, V. Azizi, X. Chen, and Y. Zhang, "Learning heterogeneous knowledge base embeddings for explainable recommendation," *Algorithms*, vol. 11, no. 9, p. 137, 2018.

[178] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 5329–5336, Honolulu, Hawaii, USA, 2019.

[179] S. Geng, Z. Fu, J. Tan, Y. Ge, G. De Melo, and Y. Zhang, "Path language modeling over knowledge graphsfor explainable recommendation," in *Proceedings of the ACM Web Conference 2022*, vol. 1, pp. 946–955, Virtual Event, Lyon France, 2022.

[180] D. A. Van Veldhuizen, G. B. Lamont, *et al.*, "Evolutionary computation and convergence to a pareto front," in *Proceedings of the late breaking papers at the genetic programming 1998 conference*, vol. 1, pp. 221–228, Citeseer, 1998.

[181] D. Li, W.-Y. Chiu, H. Sun, and H. V. Poor, "Multiobjective optimization for demand side management program in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1482–1490, 2018.

[182] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes," *Scientific data*, vol. 2, no. 1, pp. 1–14, 2015.

[183] X. Chen, S. Jia, and Y. Xiang, "A review: Knowledge reasoning over knowledge graph," *Expert Systems with Applications*, vol. 141, no. 1, p. 112948, 2020.

[184] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 29, pp. 1–10, Austin, Texas, USA, 2015.

[185] B. Shah and H. Bhavsar, "Time complexity in deep learning models," *Procedia Computer Science*, vol. 215, no. 1, pp. 202–210, 2022.

[186] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications*, vol. 1, no. 1, pp. 43–55, 2019.

[187] J. Jazzbin, "Geatpy: the genetic and evolutionary algorithm toolbox with high performance in python," *geatpy. com. http://www. geatpy. com/(accessed 2020)*, vol. 1, no. 1, p. 1, 2020.

[188] H. Zenil, "A review of methods for estimating algorithmic complexity: options, challenges, and new directions," *Entropy*, vol. 22, no. 6, p. 612, 2020.

[189] S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis.," in *Proceedings of the ISMIR*, vol. 53, pp. 537–543, Suzhou, China, 2017.

[190] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Cañizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, and R. Palma-Behnke, "Trends in microgrid control," *IEEE Transactions on smart grid*, vol. 5, no. 4, pp. 1905–1919, 2014.

[191] M. Marzband, S. S. Ghazimirsaeid, H. Uppal, and T. Fernando, "A real-time evaluation of energy management systems for smart hybrid home microgrids," *Electric Power Systems Research*, vol. 143, no. 1, pp. 624–633, 2017.

[192] Z. Iqbal, N. Javaid, S. Iqbal, S. Aslam, Z. A. Khan, W. Abdul, A. Almogren, and A. Alamri, "A domestic microgrid with optimized home energy management system," *Energies*, vol. 11, no. 4, p. 1002, 2018.

[193] Z. Qin, D. Liu, H. Hua, and J. Cao, "Privacy preserving load control of residential microgrid via deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 12, no. 5, pp. 4079–4089, 2021.

[194] M. Farrokhabadi, S. König, C. A. Cañizares, K. Bhattacharya, and T. Leibfried, "Battery energy storage system models for microgrid stability analysis and dynamic simulation," *IEEE Transactions on Power Systems*, vol. 33, no. 2, pp. 2301–2312, 2017.

[195] S. Xia, X. Luo, K. W. Chan, M. Zhou, and G. Li, "Probabilistic transient stability constrained optimal power flow for power systems with multiple correlated uncertain wind generations," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 3, pp. 1133–1144, 2016.

[196] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[197] A. Ruano, A. Hernandez, J. Ureña, M. Ruano, and J. Garcia, "Nilm techniques for intelligent home energy management and ambient assisted living: A review," *Energies*, vol. 12, no. 11, p. 2203, 2019.

[198] A. J. Pimm, T. T. Cockerill, and P. G. Taylor, "The potential for peak shaving on low voltage distribution networks using electricity storage," *Journal of Energy Storage*, vol. 16, no. 1, pp. 231–242, 2018.

[199] NASA, "PV data in Durham." [Online] `https://power.larc.nasa.gov/data-access-viewer//`. Accessed November 16, 2021.

[200] European Commission, "PV data in Durham, 2005-2016." [Online] `https://re.jrc.ec.europa.eu/pvg-tools/en/tools.html/`. Accessed November 16, 2021.

[201] energy-stats.uk, "Electricity price data, 2020." [Online] https://energy-stats.uk/download-historical-pricing-data/. Accessed November 16, 2021.

[202] National grid ESO, "Carbon emission data, 2017-2020." [Online] https://carbonintensity.org.uk/collapseData. Accessed November 16, 2021.

[203] P. Prisecaru, "The war in ukraine and the overhaul of eu energy security.," *Global Economic Observer*, vol. 10, no. 1, p. 1, 2022.

[204] IEA, "The impact of covid-19 on electricity prices." [Online] https://www.iea.org/reports/covid-19-impact-on-electricity. Accessed November 28, 2023.

[205] N. T. Mbungu, R. C. Bansal, R. M. Naidoo, and M. W. Siti, "Analysis of a grid-connected battery energy storage based energy management system," in *Proceedings of the 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*, Chhattisgarh, India.

[206] X. Fang, Q. Zhao, J. Wang, Y. Han, and Y. Li, "Multi-agent deep reinforcement learning for distributed energy management and strategy optimization of microgrid market," *Sustainable cities and society*, vol. 74, no. 1, p. 103163, 2021.

[207] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 1, pp. 279–292, 1992.

[208] G.-F. Angelis, C. Timplalexis, S. Krinidis, D. Ioannidis, and D. Tzovaras, "Nilm applications: Literature review of learning approaches, recent developments and challenges," *Energy and Buildings*, vol. 261, no. 1, p. 111951, 2022.

[209] H. Abdi, "The kendall rank correlation coefficient," *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA*, vol. 1, no. 1, pp. 508–510, 2007.

[210] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[211] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, vol. 1, no. 1, pp. 1–15, 2014.

[212] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[213] J. Hauke and T. Kossowski, "Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data," *Quaestiones geographicae*, vol. 30, no. 2, pp. 87–93, 2011.

[214] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *computer science*, vol. 1, no. 1, pp. 338–342, 2014.

[215] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[216] UCI, "Individual household electric power consumption dataset, 2006-2010." [Online] `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption`. Accessed November 14, 2022.

APPENDIX A

Other simulation results

A.1 Recommendation system privacy leakage

Apart from the scenario in Chapter 5 where the training iterations are 1000 and the batch size is 2, the recommendation system's other conditions are also listed here. The trend of changes is consistent with that in Fig. 5.13.

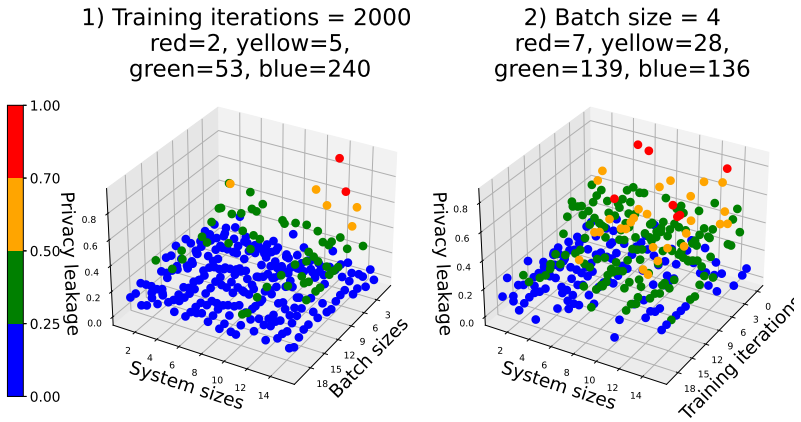

Figure A.1: The influence of different system structures. 1. System size and batch size trends when the number of training iterations is 2000. 2. System size and number of training iterations trends when the batch size is 4.

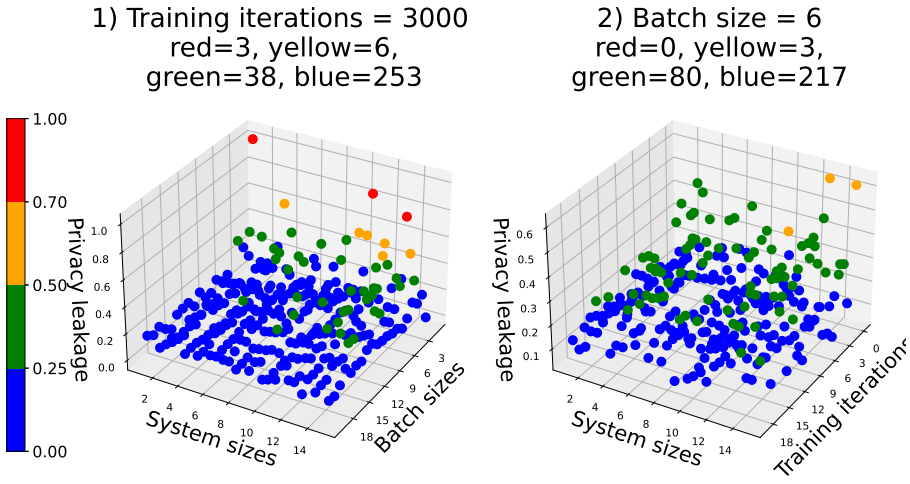

Figure A.2: 1. System size and batch size trends when the number of training iterations is 3000. 2. System size and number of training iterations trends when the batch size is 6.

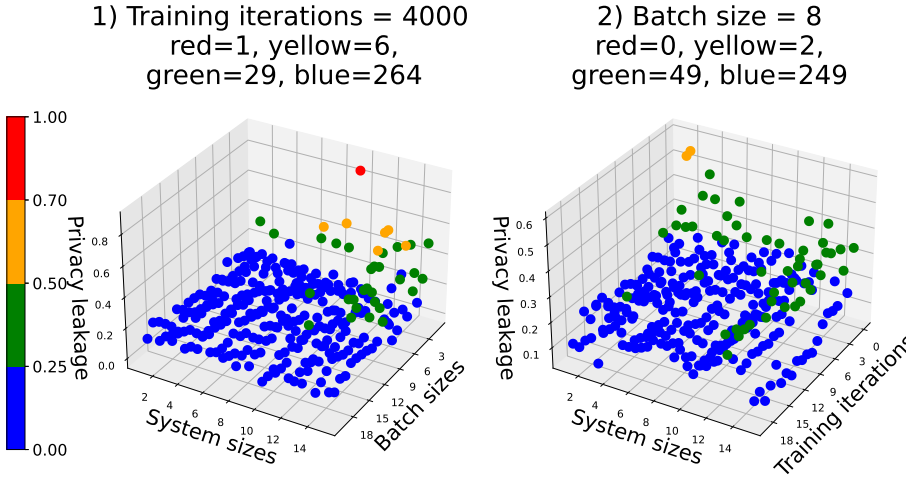

Figure A.3: 1. System size and batch size trends when the number of training iterations is 4000. 2. System size and number of training iterations trends when the batch size is 8.

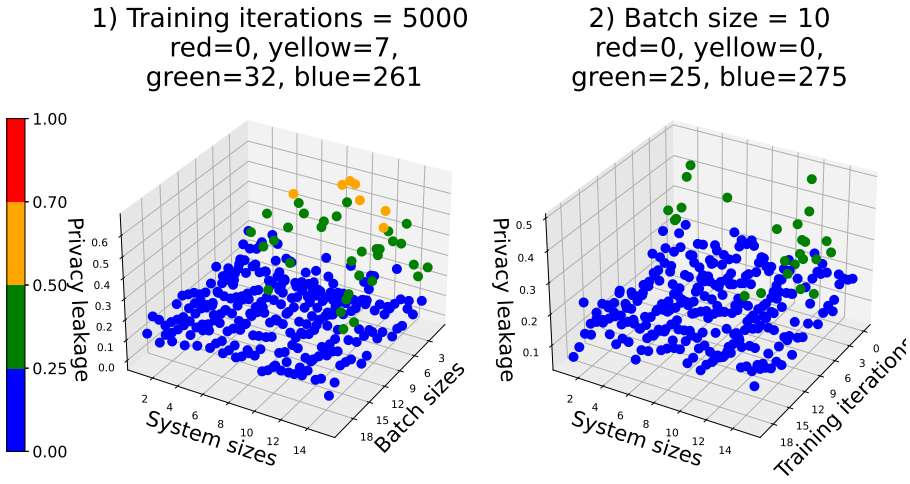

Figure A.4: 1. System size and batch size trends when the number of training iterations is 5000. 2. System size and number of training iterations trends when the batch size is 10.

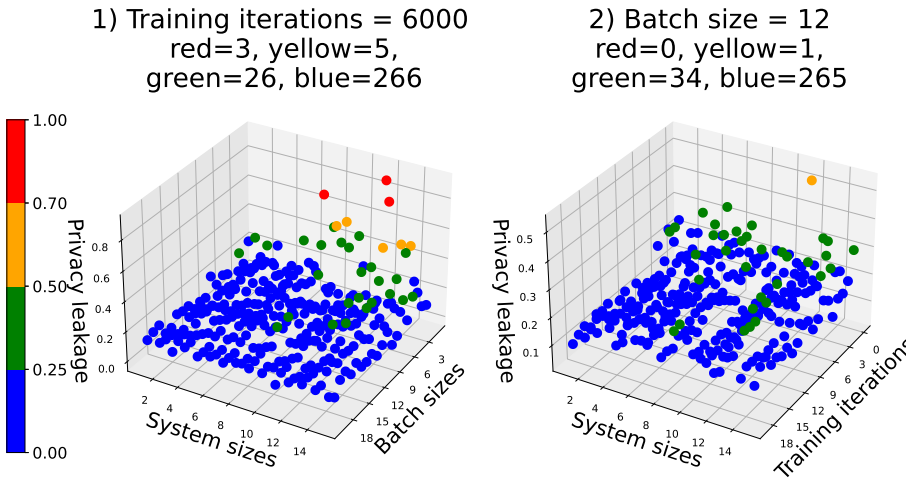

Figure A.5: 1. System size and batch size trends when the number of training iterations is 6000. 2. System size and number of training iterations trends when the batch size is 12.

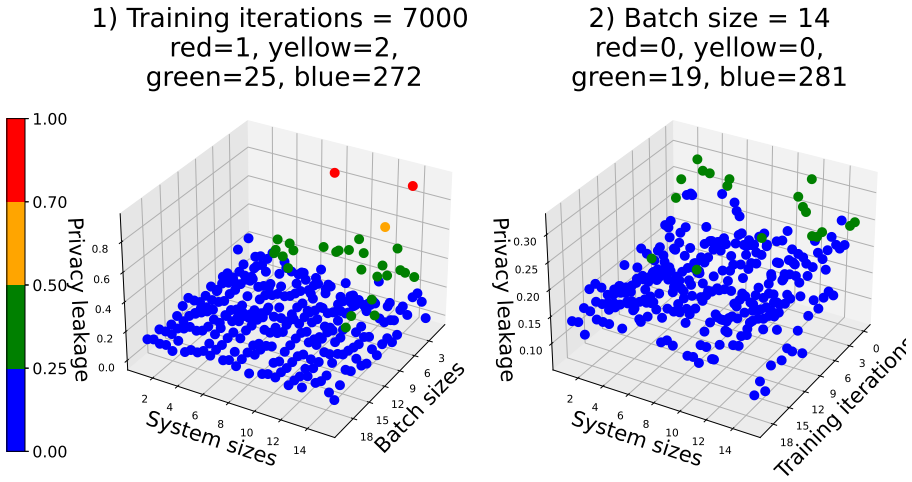

Figure A.6: 1. System size and batch size trends when the number of training iterations is 7000. 2. System size and number of training iterations trends when the batch size is 14.

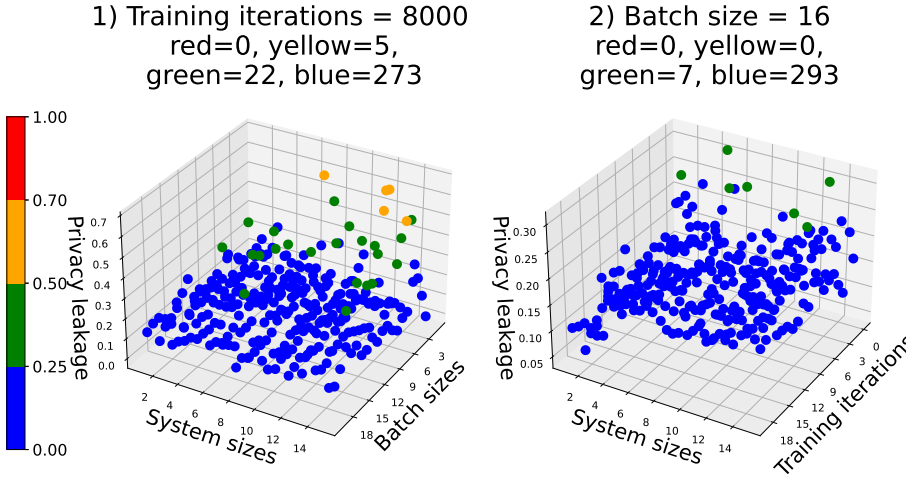

Figure A.7: 1. System size and batch size trends when the number of training iterations is 8000. 2. System size and number of training iterations trends when the batch size is 16.

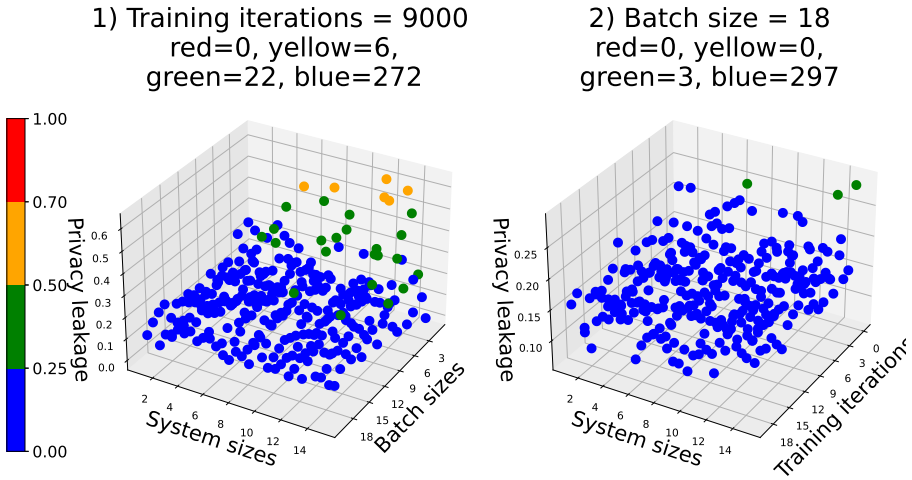

Figure A.8: 1. System size and batch size trends when the number of training iterations is 9000. 2. System size and number of training iterations trends when the batch size is 18.

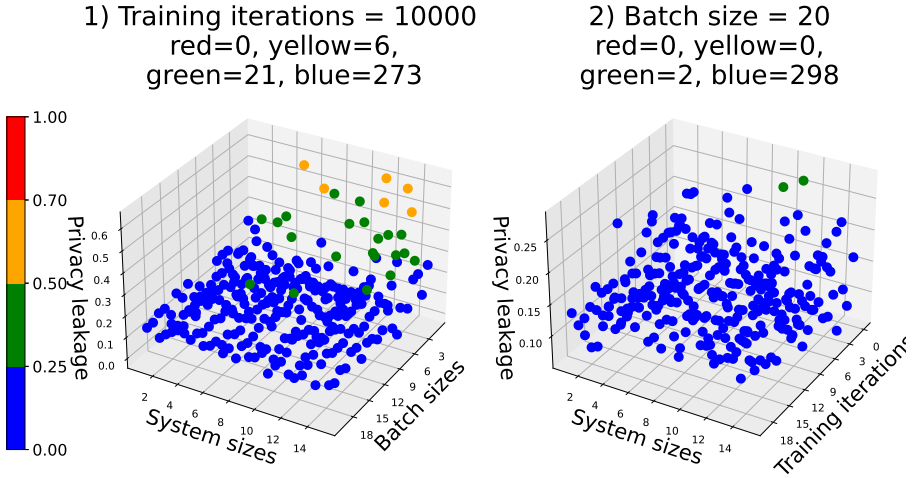

Figure A.9: 1. System size and batch size trends when the number of training iterations is 10000. 2. System size and number of training iterations trends when the batch size is 20.

A.2 Prediction system privacy leakage

Apart from the scenario in Chapter 5 where the training iterations are 5000 and the batch size is 10, the recommendation system's other conditions are also listed here. The trend of changes is consistent with that in Fig. 5.14.

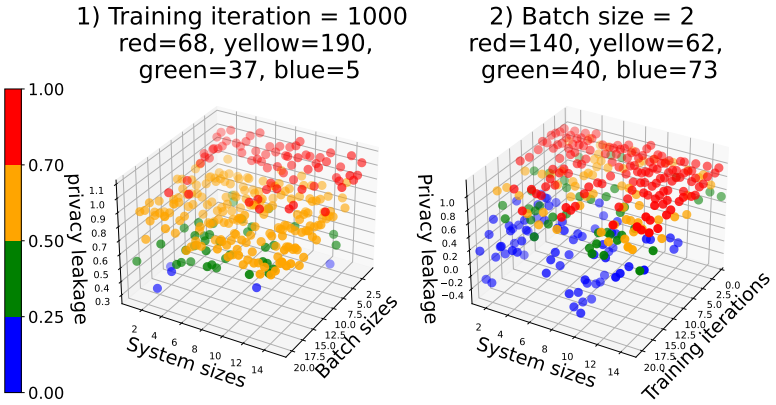

Figure A.10: 1. The trend of system size and batch size when training iterations is 1000. 2. The changes in system size and training iterations when batch size is 2.

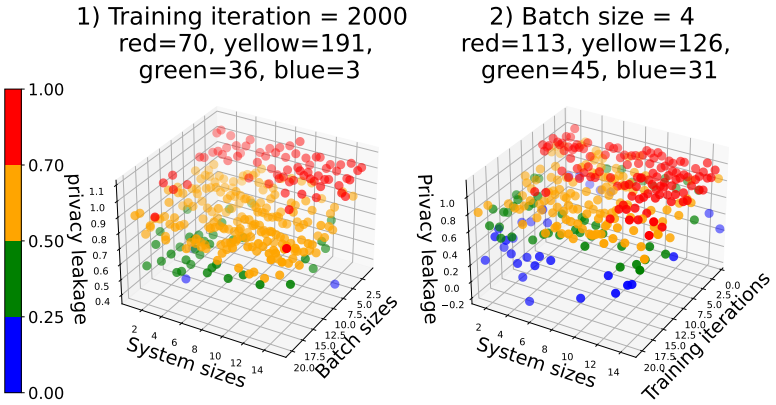

Figure A.11: 1. The trend of system size and batch size when training iterations is 2000. 2. The changes in system size and training iterations when batch size is 4.

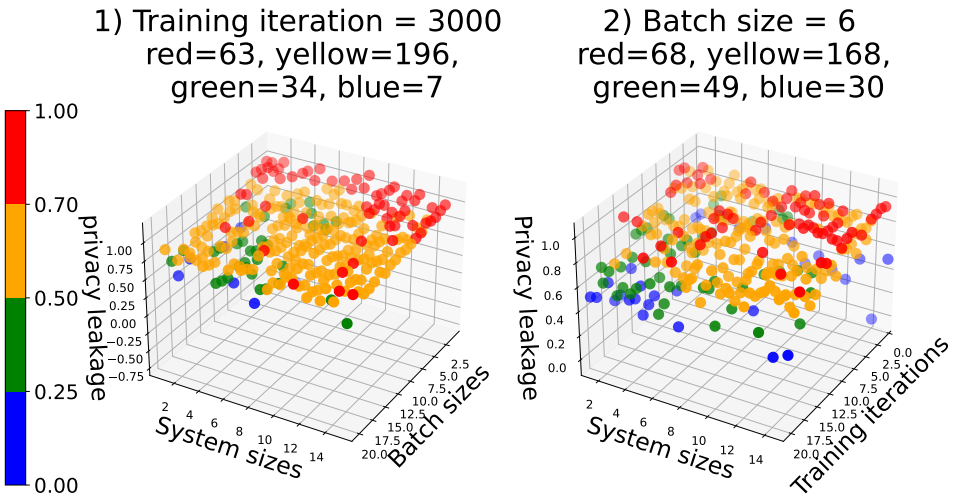

Figure A.12: 1. The trend of system size and batch size changes when the number of training iterations is 3000. 2. The changes in system size and number of training iterations when the batch size is 6.

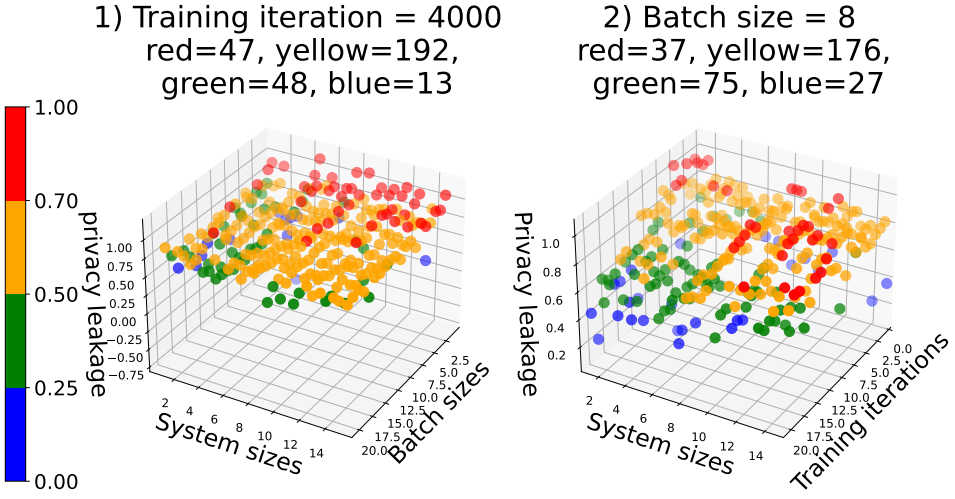

Figure A.13: 1. The trend of system size and batch size changes when the number of training iterations is 4000. 2. The changes in system size and number of training iterations when the batch size is 8.

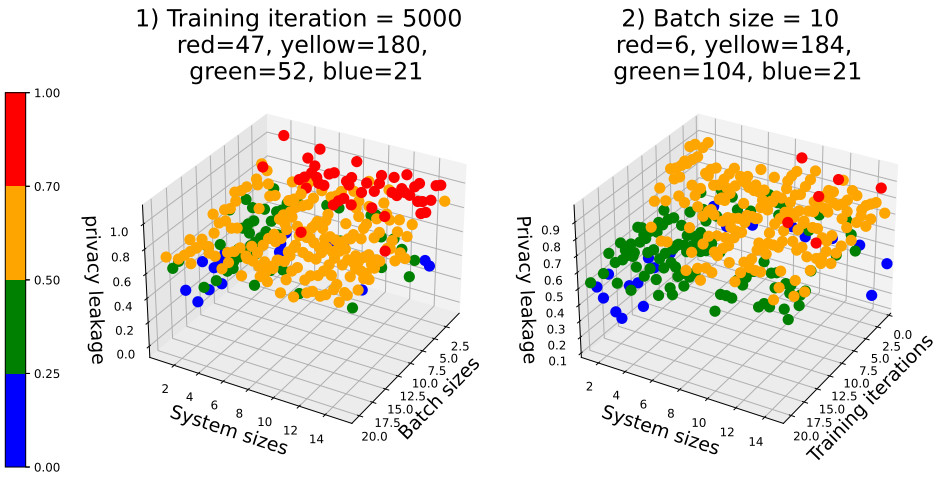

Figure A.14: 1. The trend of system size and batch size changes when the number of training iterations is 5000. 2. The changes in system size and number of training iterations when the batch size is 10.

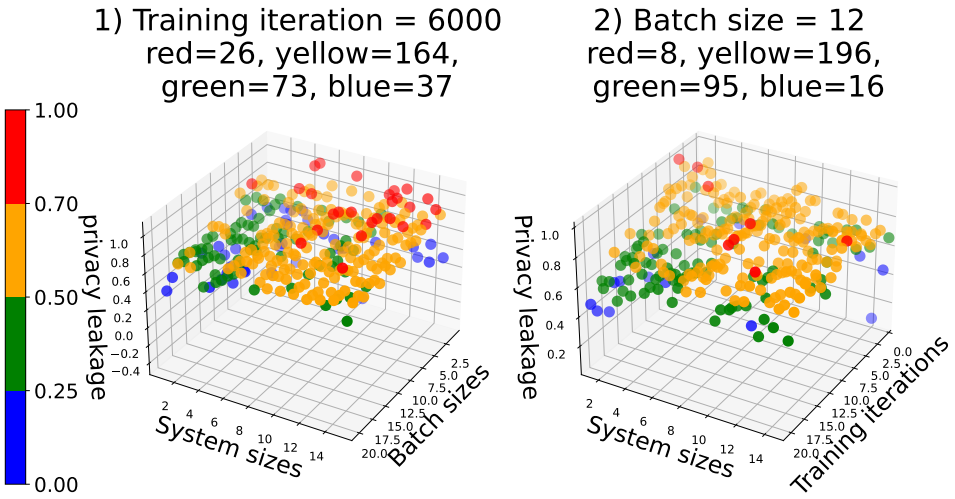

Figure A.15: 1. The trend of system size and batch size changes when the number of training iterations is 6000. 2. The changes in system size and number of training iterations when the batch size is 12.

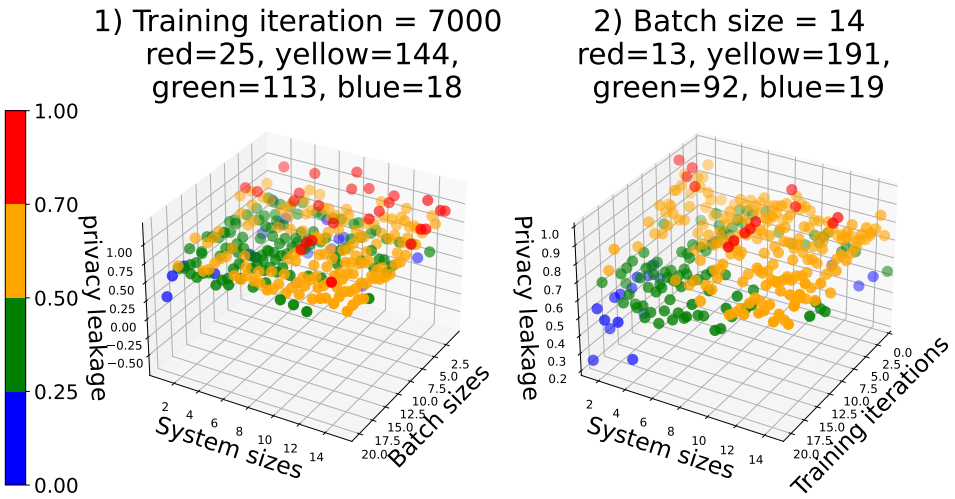

Figure A.16: 1. The trend of system size and batch size changes when the number of training iterations is 7000. 2. The changes in system size and number of training iterations when the batch size is 14.

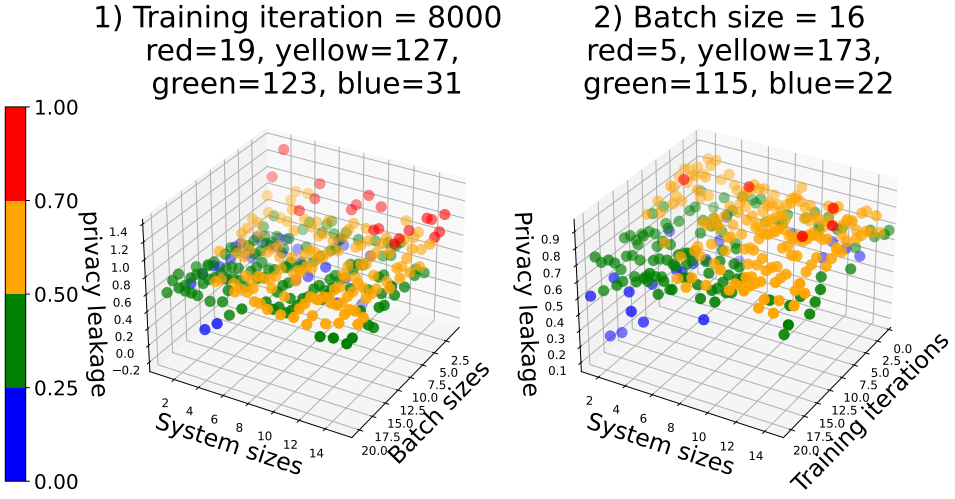

Figure A.17: 1. The trend of system size and batch size changes when the number of training iterations is 8000. 2. The changes in system size and number of training iterations when the batch size is 16.

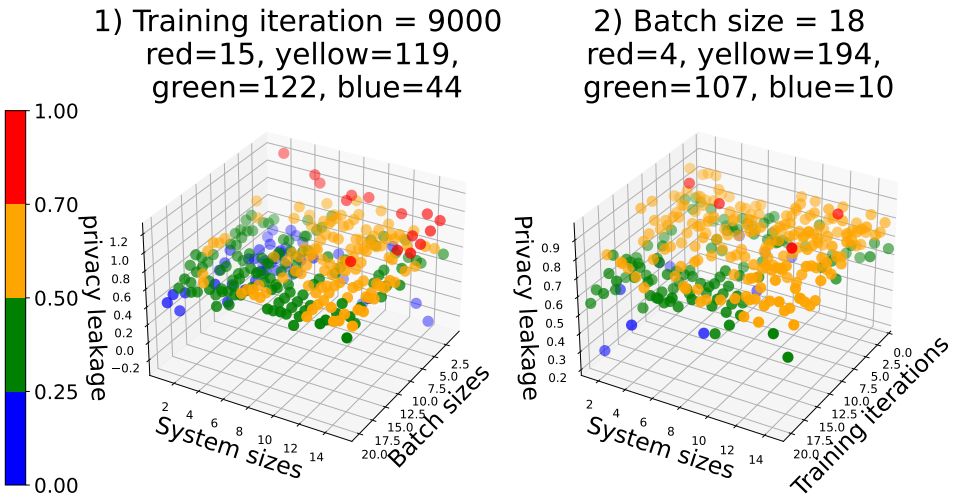

Figure A.18: 1. The trend of system size and batch size changes when the number of training iterations is 9000. 2. The changes in system size and number of training iterations when the batch size is 18.

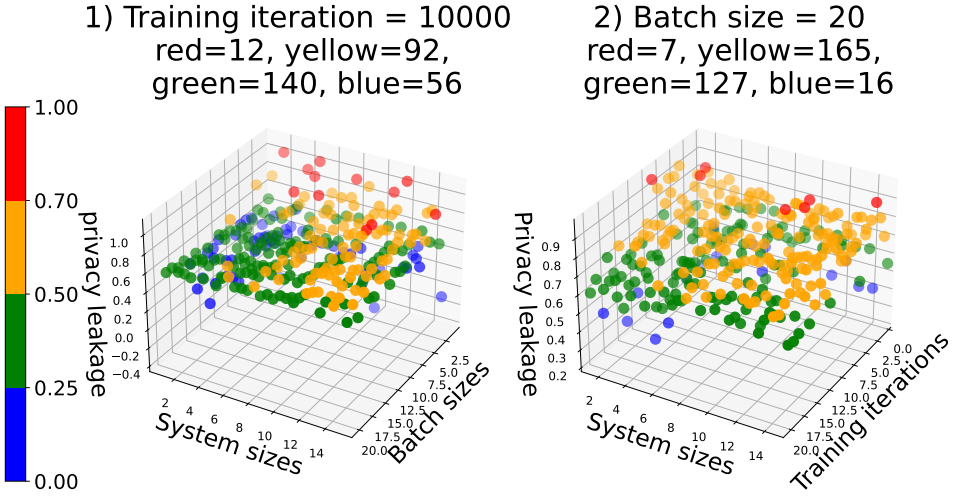

Figure A.19: 1. The trend of system size and batch size changes when the number of training iterations is 10000. 2. The changes in system size and number of training iterations when the batch size is 20.