University of Rhode Island

DigitalCommons@URI

Open Access Dissertations

2020

DEEP LEARNING FOR DATA-DRIVEN DECISION-MAKING

Zhiqiang Wan

University of Rhode Island, zhiqiang_wan@uri.edu

Follow this and additional works at: https://digitalcommons.uri.edu/oa_diss

Recommended Citation

Wan, Zhiqiang, "DEEP LEARNING FOR DATA-DRIVEN DECISION-MAKING" (2020). *Open Access Dissertations*. Paper 1209.

https://digitalcommons.uri.edu/oa_diss/1209

This Dissertation is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.

DEEP LEARNING FOR DATA-DRIVEN DECISION-MAKING

BY

ZHIQIANG WAN

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2020

DOCTOR OF PHILOSOPHY DISSERTATION
OF
ZHIQIANG WAN

APPROVED:

Dissertation Committee:

Major Professor Haibo He

Yan Sun

Lisa DiPippo

Brenton DeBoef

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2020

ABSTRACT

As we are entering the big data era, data are becoming high-volume, complex, and heterogeneous. Nowadays, almost every field and sector of the modern society is being impacted by big data, ranging from energy system to health care, and from business to government. The excessive amount of data contains potential and highly useful values. Extracting values from these data and utilize it to support the decision-making process is of paramount importance to improve productivity in business and enhance the well-being of our society. However, data-driven decision-making also arises with many challenges, such as feature selection, data fusion, and real-time decision-making.

To overcome these challenges, this dissertation will develop deep learning algorithms and frameworks for data-driven decision-making. This work is composed of three major parts: automatic feature extraction, heterogeneous data fusion, and deep reinforcement learning (DRL) based data-driven frameworks. This dissertation will first investigate the automatic feature extraction to avoid the burden of the “feature engineering” process. To make sure the feature extractor can be deployed to different scenarios, domain adaptive feature extraction will be studied. This dissertation will then examine the heterogeneous data fusion problem where feature refinement, normalization, transformation, and fusion will be investigated. With the automatic feature extraction and heterogeneous data fusion, this dissertation will develop data-driven decision-making frameworks for real-world applications, including robot-assisted pedestrian regulation and real-time electric vehicle (EV) charging management. Numerous experiments will be conducted to verify the effectiveness of the proposed frameworks.

ACKNOWLEDGMENTS

During my study at the University of Rhode Island (URI), I have received a great deal of support and inspiration from a number of wonderful individuals. I am forever thankful to all of them for being part of this journey and making this dissertation possible. The first person I want to thank is my advisor, Dr. Haibo He, who convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without his persistent help, this work would not have been completed.

I also wish to express my sincere appreciation to Dr. Yan (Lindsay) Sun, Dr. Lisa DiPippo, and Dr. Marco Alvarez for their invaluable time and suggestions for my research. I have benefited immensely from their knowledge and experience.

I would also like to thank many exceptional fellow colleagues in the Computational Intelligence and Self-Adaptive System (CISA) group. Thank you, Zhen, Yufei, Bo, Jing, Jun, Xiangnan, He, Yazhou, Lusi, Zhenhua, Hepeng, Fanrong for your feedback and support on various aspects of this work. I am really thankful for other fellow visiting scholars at the CISA lab for their help over the past five years.

I wish to show my deep and sincere gratitude to my parents for their continuous and unparalleled love and help. Last but not least, I am forever indebted to my wife, Cheng, for her love and great support when I pursue my doctoral degree.

I am also deeply grateful to the support for my Ph.D. research from the National Science Foundation (NSF) under grants ECCS 1053717, CCF 1439011, CMMI 1526835, ECCS 1731672 and ECCS 1917275, and the Office of Naval Research under grant N00014-18-1-2396. Their support provided me the opportunity to explore the important research topics as presented in this dissertation.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	x
CHAPTER	
1 Introduction	1
1.1 Research Motivations and Challenges	1
1.2 Deep Learning	2
1.2.1 Deep Neural Network	3
1.2.2 Deep Reinforcement Learning	8
1.3 Dissertation Outline	10
2 Deep Learning for Automatic Feature Extraction	13
2.1 Chapter Overview	13
2.2 Supervised Feature Extraction	14
2.2.1 Domain Adaptation	15
2.2.2 Domain Adaptive Feature Extractor	17
2.2.3 Experimental Results	20
2.3 Unsupervised Feature Extraction	30
2.3.1 Autoencoder	30

	Page
2.3.2 Generative Model for Sparse Autoencoder	32
2.3.3 Experimental Results	38
2.4 Chapter Summary	45
3 Heterogeneous Data Fusion for Decision-Making	48
3.1 Chapter Overview	48
3.2 Background and Related Works	49
3.3 Feature Fusion for Heterogeneous Data	51
3.3.1 Attention Network	52
3.3.2 Fusion Network	54
3.4 Experimental Results	57
3.4.1 Experimental Setup	57
3.4.2 Evaluation Results on VQA Dataset	59
3.5 Chapter Summary	62
4 Data-Driven Decision-Making for Robot-Assisted Pedestrian Regulation	64
4.1 Chapter Overview	64
4.2 Background and Related Works	65
4.2.1 Human-Robot Interaction	65
4.2.2 Reinforcement Learning for Robot Control	66
4.3 Problem Formulation	68
4.4 Data-Driven Robot Motion Control	71
4.4.1 Overall Architecture	71
4.4.2 Training Process	73

	Page
4.4.3 Online Deployment	75
4.5 Experimental Results	76
4.5.1 Experimental Setup	76
4.5.2 HRI Characteristics	79
4.5.3 Performance Evaluation	82
4.5.4 Analysis of the Extracted Hidden Features	94
4.6 Chapter Summary	97
5 Data-Driven Decision-Making for EV Charging Management	98
5.1 Chapter Overview	98
5.2 Background and Related Works	98
5.3 Problem Formulation	101
5.4 Data-Driven EV Charging Management	103
5.5 Experimental Results	107
5.5.1 Experimental Setup	108
5.5.2 Performance Evaluation	110
5.5.3 Ablation Studies	115
5.6 Chapter Summary	117
6 Conclusions	119
6.1 Summary of the Dissertation	119
6.2 Challenges and Opportunities	121
LIST OF REFERENCES	123
BIBLIOGRAPHY	138

LIST OF FIGURES

Figure		Page
1	Convolution operation in CNN.	4
2	An unfolded LSTM network.	6
3	The structure of the LSTM cell.	7
4	A typical agent-environment scheme in DRL.	9
5	Overall structure of the dissertation.	11
6	The overall architecture of the proposed OSFA.	19
7	Architecture of autoencoder.	30
8	Relationship between the average activation and the coefficient β	37
9	Features learned by sparse autoencoder with 169 hidden units.	42
10	Activation of hidden units in autoencoder and sparse autoencoder.	43
11	Features learned by sparse autoencoder with 196 hidden units.	46
12	An example for the VQA task.	51
13	Overall architecture of the proposed AnswerNet.	51
14	Architecture of the attention network.	53
15	Architecture of the hierarchical fusion network.	55
16	Snapshots of the generated answers.	60
17	The merging pedestrian flows scenario.	69
18	The overall framework of the proposed data-driven robot motion control for pedestrian regulation.	72
19	Snapshot of the environment used in the simulation.	77
20	HRI characteristics for Case 1.	80

Figure		Page
21	HRI characteristics for Case 2.	80
22	HRI characteristics for Case 3.	81
23	The evolution of the accumulated outflow over 3500 epochs during the training process.	83
24	Robot initial position [0.4, 2] for Case 1.	84
25	Robot initial position [3.4, 7.4] for Case 1.	85
26	Average accumulated outflow for Case 1 under robot initial positions [0.4, 2] and [3.4, 7.4] over 10 runs.	87
27	Time history of robot position under 10 robot initial positions for Case 1.	87
28	Robot initial position [0.4, 2] for Case 2.	88
29	Robot initial position [3.4, 7.4] for Case 2.	89
30	Average accumulated outflow for Case 2 under robot initial positions [0.4, 2] and [3.4, 7.4] over 10 runs.	90
31	Time history of robot position under 10 robot initial positions for Case 2.	91
32	Evaluation results for Case 3.	93
33	The 3D environment constructed by Unity 3D engine.	94
34	Evaluation results in the 3D environment.	95
35	Two-dimensional t-SNE embedding of the hidden representations of the Q network.	96
36	The Overall architecture of the proposed data-driven EV charging management.	105
37	The evolution of the cumulative rewards during the training process.	110
38	Cumulative charging cost of the proposed and benchmark solutions over the 120 test days.	112

Figure		Page
39	Charging/discharging patterns of the proposed approach over 7 consecutive days.	113
40	Average range anxiety (blue circle) and charging cost (red square) under different anxiety coefficient.	114
41	Daily charging cost and battery SOC when the EV leaves home under different anxiety coefficient τ	116

LIST OF TABLES

Table		Page
1	Adapt from <i>Cityscapes</i> to <i>Foggy Cityscapes</i>	22
2	Adapt from <i>SIM 10k</i> to <i>Cityscapes</i>	24
3	Adapt from <i>SIM 10k</i> to <i>PASCAL VOC</i>	25
4	Feature alignment with different feature layers.	26
5	Domain adaptation with different target sample.	27
6	Domain adaptation with different coefficient α	28
7	Average activation of hidden units on MNIST dataset	39
8	Classification accuracy of sparse autoencoder with 169 hidden units.	41
9	Classification accuracy of sparse autoencoder with 196 hidden units.	45
10	Evaluation results on VQA <i>test-dev</i>	61
11	Evaluation results on VQA <i>test-standard</i>	62
12	Evaluation results on VQA2 <i>test-standard</i>	63
13	Computational time and resource usage	92
14	Random variables for commuting behavior.	109

CHAPTER 1

Introduction

1.1 Research Motivations and Challenges

There is no doubt that we are entering the big data era. According to the computer giant IBM, every day, we produce more than 2.5 billion GB data [1]. There are 3.5 billion requests on Google and 152 million purchases on Amazon per day. Facebook generates 500 terabytes data, and Twitter generates 12 terabytes data every day. In addition, the speed of data growth has already exceeded Moore's law [2]. IBM also reported that about 90% of all the data in the world today has been created in the past two years.

With this explosion in data and the rapid development of information technology, decision makers are gaining improved capability to extract value from the data and use it to support the decision-making process [3]. Now, data lies at the heart of decision-making process. Consequently, data-driven decision-making has received considerable attention from both the academia and industry [4, 5, 6, 7, 8]. Traditional model-based decision-making, which requires intensive human involvement, is inefficient and contains human bias. Real-world systems are becoming more and more complex. These systems generally contain hundreds of components and numerous measurement units. These systems are becoming really overwhelming for traditional model-based decision-making. Data-driven decision-making has the potential to solve these problems and can greatly reduce human intervention and improve the performance and efficiency.

Data-driven decision-making can be briefly divided into three major steps. The first step is to select suitable feature representations from the data. Feature selection is an important step to reduce the complexity of the input data [9, 10, 11]. Then, if the data come from different sources, the data should be properly

fused. Data fusion is required to capture the underlying relationships in these data [12, 13, 14]. Finally, the fused features will be further analyzed to support the decision-making process [3].

The emergence of big data phenomenon has given rise to datasets that are massive, highly heterogeneous, and dynamically evolving, as well as incomplete, noisy and imprecise [15, 16, 17]. These distinct characteristics pose great challenges for data-driven decision-making [18]. First, special domain knowledge is required to select or design features to reduce the complexity of the input data. This process is called feature engineering, which is generally time-consuming and labor-intensive. For example, Scale Invariant Feature Transform (SIFT) [19] and Histogram of Oriented Gradients (HOG) [20] are widely used feature engineering algorithms for computer vision. Performing automatic feature extraction would be a major breakthrough for data-driven decision-making. Another challenge is data heterogeneity. The heterogeneity and complexity of the data call for advanced algorithms that can capture the underlying relationship and allow reconciling the data heterogeneity arising due to diverse data types. The last challenge is how to transform the data into actionable insights and enable automated decision making and control that requires autonomous agents to interact with the real world.

1.2 Deep Learning

Among many efforts towards data-driven decision-making, deep learning provides the key technical innovations [21, 22, 23]. For instance, deep neural network (DNN) can effectively and automatically extract discriminative features from input data, which would greatly benefit data-driven decision-making. In addition, DNN has demonstrated its strong capability in transforming heterogeneous data into common feature space and fusing these data. Moreover, deep reinforcement learning (DRL) holds the promise of automating complex decision-making and

control in the presence of uncertainty. For a wide range of real-world problems, from robotic control and autonomous vehicle to dynamic energy management system, DRL would provide dramatic improvement in capability and reduction in engineering cost.

1.2.1 Deep Neural Network

Deep neural network has a layered, hierarchical architecture and can automatically extract discriminative feature representations from the input data [24, 25, 26]. For example, given a face image, the lower layers of DNN can extract low-level features, such as edges and corners. The middle layers can extract middle-level features, such as eyes, noses, and mouths. The upper layers can detect the whole face. The more layers DNN has, the better features it can extract. Network trained by the Group Method of Data Handling in 1960s was perhaps the first DNN system [27]. However, it was not until 2010s that DNN started to be widely used. The booming of DNN is driven by three major forces. First, the availability of big data makes it possible to train a DNN without worrying about the over-fitting problem. Second, high-performance computing, like GPU, greatly reduces the training time. Third, the development of advanced training algorithms makes the training process converge to good optimal point instead of being trapped by the saddle points or local minimums. Recently, many powerful DNN architectures have been developed and obtained great success in many complex applications, such as computer vision [28, 29, 30, 31, 32], natural language processing [33, 34, 35, 36], and speech recognition [37, 38, 39]. This section will introduce two types of DNNs that are widely used in data-driven decision-making: one is convolutional neural network (CNN) [40]; the other is long short-term memory (LSTM) network [41]. CNN has achieved great success in processing spatial correlated data while LSTM network is widely used in time-series analysis.

Convolutional Neural Network

The architecture of CNN is inspired by cat's visual cortex. Hubel et al. found that cat's visual cortex has a complex arrangement of cells [42]. Inspired by this arrangement, CNN is built up with convolutional layers and pooling layers. In recent years, many deep CNNs, such as AlexNet [28], GoogLeNet [43], VGGNet [44], and ResNet [45], have been proposed and revolutionized computer vision domain. Fig. 1 [40] shows the convolution operation in the convolutional layer. The unit on the hidden layer is connected to a region of the input layer via the convolutional kernel. This region is called local receptive field. In Fig. 1, the size of kernel is 5×5 . The kernel will slide over the input layer from left to right and from top to bottom. In Fig. 1, the stride size is 1. So, the output of hidden unit at position (j, k) is calculated by Eq.(1).

Figure 1: Convolution operation in CNN.

$$y_{j,k} = f \left(b + \sum_{l=0}^4 \sum_{m=0}^4 w_{l,m} x_{j+l,k+m} \right) \quad (1)$$

where w is the weight and b is the bias of the convolutional kernel; x represents the input layer; $f(\cdot)$ is the activation function. The output of this convolutional operation is called feature map. Each feature map can extract one specific kind

of feature. In order to extract different kinds of features, multiple convolutional kernels will be used to generate multiple feature maps. The details regarding the convolution operation can be found in [46].

Fig. 1 also shows some important properties of the convolutional layer. First, the convolutional kernel only connects to a small region instead of the whole input layer. This sparse connectivity property can greatly reduce the computational cost. Second, the weight of the convolutional kernel is shared during the convolution process. Compared with the fully-connected neural network, the number of weights is significantly reduced. Third, the convolution operation is translation invariant. In other words, the convolutional kernel can detect the input pattern no matter where the input pattern is.

Pooling layer is generally used after the convolutional layer to condense the output of the previous convolutional layer. The output is calculated by pooling a non-overlapping region in the input layer. Max pooling and average pooling are typically used as the pooling operation. For max pooling, the output is the maximum activation of the input region. For average pooling, the output is the average activation of the input region.

The pooling operation reduces the dimension of the input layer and therefore reduces the computational operations. In addition, it makes CNN invariant to translation in the small region. Specifically, if the input pattern appears in any position of the pooled region, the output will be the same.

Long Short-Term Memory Network

LSTM network is a type of recurrent neural network (RNN) [47]. It is proposed to alleviate the gradient vanishing problem in RNN. The idea behind the LSTM network is to make use of sequential information. LSTM network performs the same processing for every element of the sequence, with the output being dependent

on the previous computations. The information about what has been calculated so far can be stored or “memorized” in the LSTM cells. The typical structure of an LSTM network is shown in Fig. 2 [41].

Figure 2: An unfolded LSTM network.

Fig. 2 shows an LSTM network being unfolded into a full network. Unfolding means that we write out the network for the complete sequence. In Fig. 2, d_t is the input at the current step; c_{t-1} and y_{t-1} are the cell state and output from previous time step.

The structure of the LSTM cell is shown in Fig. 3 [41]. The key to the LSTM network is the cell state c_t . The LSTM network has the ability to add information into or remove information from the cell state, carefully regulated by structures called gates. Gates provide a way to optionally let information through. Specifically, an input gate determines the amount of information to be added into the cell state while a forget gate determines the amount of information to be inherited from the previous cell state c_{t-1} . The input gate and forget gate are shown in Eq. (2) and Eq. (3), respectively,

$$i_t = \sigma(W_i * d_t + R_i * y_{t-1} + b_i) \quad (2)$$

$$f_t = \sigma(W_f * d_t + R_f * y_{t-1} + b_f), \quad (3)$$

Figure 3: The structure of the LSTM cell.

where W_i , R_i , W_f , and R_f are the weights for the input gate and forget gate; b_i , and b_f are the biases for these gates; σ is the sigmoid function whose output is between 0 and 1, describing how much of information should be let through.

The input information z_t is shown as

$$z_t = h(W_z * d_t + R_z * y_{t-1} + b_z), \quad (4)$$

where h denotes the hyperbolic tangent function; W_z and R_z are the weights; b_z is the bias. The input gate would determine the amount of z_t to be added into the cell. Therefore, the cell state is calculated as

$$c_t = i_t \odot z_t + f_t \odot c_{t-1}, \quad (5)$$

where $\odot$ represents the element-wise multiplication operation; $i_t \odot z_t$ denotes the amount of information to be added from z_t ; $f_t \odot c_{t-1}$ denotes the amount of infor-

mation to be inherited from the previous cell state c_{t-1} .

The output of the cell is determined by the output gate as

$$o_t = \sigma(W_o * d_t + R_o * y_{t-1} + b_o), \quad (6)$$

where W_o , R_o are the weights, and b_o is the bias. Thus, the output of the cell is shown as

$$y_t = o_t \odot h(c_t). \quad (7)$$

1.2.2 Deep Reinforcement Learning

Much of the inspiration for RL algorithm came from rules developed to describe the learning behavior of animals [48, 49, 50]. Let us take a look at a real-world experiment: train a mouse to navigate a maze looking for food [51]. The mouse needs to make decisions. Wrong decisions result in the punishment of hunger while good decisions get the mouse to the food more quickly. At the beginning of the experiment, the mouse took a long time to find the food. Then, after repeating this experiment several times, the mouse finally found the best path to the food. For this experiment, exploring the maze is a way to collect data. With these data, the mouse learns to make good decisions. Learning from the experience data is the fundamental learning strategy of RL [52, 53].

RL establishes a framework for data-driven decision-making [52, 54]. In RL, an agent takes actions and interacts with an environment. For the previous experiment, mouse is the agent, and maze is the environment. RL has three basic elements, state, action, and reward. In this experiment, the state is the location of the mouse in the maze. The action is the move of the mouse. The reward measures the quality of the move. Wrong moves result in the punishment of hunger while good moves get the mouse to the food more quickly. After trial and error, the mouse will learn a good policy to maximize the reward. More details about RL

can be found in [52].

DNN is an architecture that mimics animal brain. It can extract high-level features from the input data and effectively fuse the data from different sources. RL is a general framework for data-driven decision making. The combination of these two different models is one of the best answers we have so far to solve challenging real world problems. This new combination is called DRL. A typical agent-environment scheme in DRL is shown in Fig. 4.

Figure 4: A typical agent-environment scheme in DRL.

The idea of DRL was first proposed by Mnih et al. [50] to train an agent to play Atari games. In the proposed framework, CNN is applied to extract suitable features from the high-dimensional input images. Then, the Q-network is used to generate actions based on these features. The overall framework is trained with

RL. The performance of DRL on the Atari games was comparable to that of a professional human player. This achievement was mostly contributed by the representation learning with DNN that enabled automatic feature extraction and end-to-end RL through gradient descent. In recent years, many DRL-based data-driven decision-making frameworks, such as AlphaGo [55], Deepstack [56], AlphaStar [57], have achieved great success in many complex applications.

1.3 Dissertation Outline

Innovative research is needed to address the growing complexity of data-driven decision-making. This dissertation focuses on the development of deep learning algorithms for data-driven decision-making. The overall structure of the dissertation is illustrated in Fig. 5. Following Chapter 1 of introduction, Chapter 2 presents supervised and unsupervised feature extractors. Then, Chapter 3 investigates the heterogeneous data fusion problem. With automatic feature extraction and heterogeneous data fusion, in Chapter 4 and Chapter 5, data-driven decision-making frameworks are developed to solve two real-world problems, robot-assisted pedestrian regulation and electric vehicle (EV) charging management. Chapter 6 concludes this dissertation with discussions on future challenges and opportunities.

Specifically, Chapter 2 first introduces the supervised feature extractor and how domain shift undermines its performance. A low-level feature alignment method is proposed to reduce the domain shift and maintain the performance of feature extractor across different domains. The experimental results under different domain adaptation scenarios verify the effectiveness of the proposed method. Chapter 2 also introduces an unsupervised feature extractor, sparse autoencoder. It does not need label information and thus does not suffer from the domain shift problem. A generative model is established to model the sparse autoencoder and determine the sparse hyperparameter. The research results presented in this chap-

Figure 5: Overall structure of the dissertation.

ter have also been reported in [58, 59].

Chapter 3 studies the heterogeneous data fusion problem, which is a crucial step in data-driven decision-making. Two typical types of heterogeneous data, image and natural language, are investigated in this chapter. An attention network is used to capture the correlation between the input data and refine the input data according to the correlation. In addition, a hierarchical fusion network is developed to transform the heterogeneous data into a common feature space. In the fusion network, a multi-head fusion is proposed to effectively fuse the data.

The experimental results on a large-scale heterogeneous dataset demonstrate the effectiveness of the proposed fusion technique. The research results presented in this chapter have also been reported in [60].

Chapter 4 develops a data-driven decision-making framework based on DRL for a real-world problem, robot-assisted pedestrian regulation. To avoid crowd accident, a mobile robot is introduced to regulate pedestrian flows. A data-driven approach is developed to find the optimal policy that maneuvers the robot to achieve the maximum pedestrian outflow. The proposed approach is evaluated under different pedestrian flow conditions and different robot initial positions. The visualization of the extracted features reveals how the automatic feature extraction contributes to the success of the proposed data-driven approach. The research results presented in this chapter have also been reported in [61].

Chapter 5 develops a data-driven solution for the EV charging problem in smart grid. The proposed data-driven approach can learn an optimal charging policy that can not only reduce the customer's charging cost but also make sure the EV can be fully charged upon departure. In this chapter, the EV charging problem is formulated as a Markov decision process (MDP) where the uncertainty of the future electricity prices and user's commuting behavior are considered. Numerous simulations are conducted to validate the effectiveness of the proposed approach. The research results presented in this chapter have also been reported in [62].

Finally, Chapter 6 summarizes the dissertation and discusses future challenges and opportunities along the research direction.

CHAPTER 2

Deep Learning for Automatic Feature Extraction

2.1 Chapter Overview

Extracting good feature representation is the foundation of data-driven decision-making. For traditional methods, given the input data, suitable features should be selected and designed by human experts. Although this “feature engineering” process, in general, can leverage domain knowledge to improve the learning performance, it is typically labor-intensive [63]. In addition, most of the human designed features are low-level features.

Deep learning can automatically extract high-level features from input data without human intervention. This can greatly benefit data-driven decision-making. Automatic feature extractor can be briefly divided into two categories: supervised feature extractor and unsupervised feature extractor. Supervised feature extractor can extract discriminative features from the input data with the help of the labeling information. Unsupervised feature extractor can learn the underlying pattern from the input data without labeling data.

This chapter will first introduce the supervised feature extractor and the impact of domain shift on it. Then, a novel domain adaptation method is proposed to reduce the domain shift for the supervised feature extractor. The main contributions are threefold. First, the domain shift problem in supervised feature extractor is investigated. Second, a low-level feature alignment method is proposed to reduce the domain shift. Third, the effectiveness of the proposed algorithm is validated in numerous experiments under different domain adaptation scenarios.

This chapter will also introduce an unsupervised feature extractor, sparse autoencoder, and how to determine its sparse hyperparameter. The main contributions are in twofold. First, the relationship between the sparse hyperparameter

and the sparsity of sparse autoencoder is derived. Second, an effective method is proposed to determine the sparse hyperparameter.

2.2 Supervised Feature Extraction

DNN-based supervised feature extractors, like CNN, have achieved great success in many data-driven decision-making applications [31, 32, 45, 64]. With these supervised feature extractors, machines have achieved super-human performance in a large-scale image recognition task [65]. In addition, these feature extractors have been widely used in object detectors where machines can detect different objects in an image and also provide position information for each object.

Even though these feature extractors have obtained promising results in data-driven decision-making, they suffer from a common problem, domain shift. In general, these feature extractors are trained on a large amount of annotated data that are collected in a source domain. After the training process, these feature extractors are deployed in a target domain. Generally, it is assumed that the distributions of the source domain and the target domain are similar. However, in practice, this assumption is not always true. For example, the source domain data may be collected in the clear weather with good visibility while the target domain data may be collected in the foggy weather. The distribution discrepancy between the source domain and the target domain is called domain shift. Due to domain shift, the performance of these models may deteriorate sharply in the target domain. A straightforward solution is to recollect the data in the target domain and retrain the feature extractor. However, it is time-consuming and labor-intensive.

2.2.1 Domain Adaptation

To reduce domain shift and obtain good performance in the target domain, many domain adaptation algorithms have been proposed. These algorithms do not need any annotation for the target domain data. They can be mainly grouped into two categories: instance reweighting and feature matching. Instance reweighting approaches attempt to identify the training source instances that are highly relevant to the target domain and exploit those reweighted instances to train a target model [66, 67]. Feature matching methods aim to align source and target domains by reducing the distribution gap of learned domain-invariant features [68, 69]. Thereinto, transfer component analysis (TCA) [70] learns some shared transfer components in a Reproducing Kernel Hilbert Space (RKHS) to reduce the discrepancy across domains. While recent studies have shown that more transferable representative features can be learned by embedding domain adaptation in the pipeline of deep representation learning [71, 72]. Currently, inspired by generative adversarial nets (GANs), adversarial domain adaptation methods through an adversarial objective play the minimax game between the source and target domains to make them indistinguishable. For example, [73] presents a unified framework, adversarial discriminative domain adaptation (ADDA), which learns two separate encoding networks that project source and target data into the same space by a domain-adversarial loss.

Unlike these previous works that require a huge amount of samples from target domain, a one-shot domain adaptation algorithm that only needs one sample from the target domain is proposed. This can greatly reduce the burden of collecting data in the new target domain. The proposed algorithm is named as one-shot feature alignment (OSFA) [58]. This section will focus on the domain adaptation for CNN since CNN is a widely feature extractor for data-driven decision-making.

Unlike the previous works that focus on aligning high-level features, the proposed algorithm aligns the low-level features of the source domain and the target domain. Specifically, the domain shift is reduced by aligning the average activation of the feature maps in the lower layer of CNN. In addition, [74] pointed out that the lower layer in a CNN can respond to low-level features, such as corners and edges. Source domain images and target domain images share common low-level features. However, due to domain shift, the illumination or visibility of the low-level features in the target images may be different from those in the source images. This difference makes CNN hard to recognize the low-level features in the target domain. Therefore, it is essential to align low-level features to reduce domain shift. In this section, the object detection task is used to verify the effectiveness of the proposed approach. The reason to choose object detection task is that it is more challenging than other tasks, like image classification. In addition, the domain adaptation algorithm developed for object detection can also be used for other tasks.

In one-shot unsupervised domain adaptation for object detection, the images in the source domain are labeled with bounding boxes, and only one unlabeled image is available in the target domain. $\mathcal{D}_S = \{(x_{S_1}, y_{S_1}), \dots, (x_{S_n}, y_{S_n})\}$ is used to represent the samples in the source domain, where $x_{S_i} \in \mathcal{X}_S$ denotes the input image, and $y_{S_i} \in \mathcal{Y}_S$ denotes the bounding box annotation. $\mathcal{D}_T = \{x_T\}$ is used to represent the sample in the target domain, where $x_T \in \mathcal{X}_T$ denotes the input image.

The marginal distribution of source domain data X_S is denoted as $\mathcal{P}(X_S)$, and the marginal distribution of target domain data X_T is denoted as $\mathcal{Q}(X_T)$. Since there exists domain shift between the source domain and the target domain, these two distributions are not equal, i.e., $\mathcal{P}(X_S) \neq \mathcal{Q}(X_T)$. The goal is to determine a transformation function $f(\cdot)$ to map the source domain images and

the target domain images into the same feature space such that the domain shift can be reduced. In this problem, a deep CNN is implemented as the transformation function. Its parameters are optimized by minimizing the detection loss and the discrepancy between $P(f(X_S))$ and $P(f(X_T))$. With this transformation function, the domain shift will be reduced, i.e., $P(f(X_S)) \approx P(f(X_T))$.

2.2.2 Domain Adaptive Feature Extractor

In this section, first, the motivations of low-level feature alignment and one-shot domain adaptation are explained. Then, the details about the proposed OSFA algorithm for one-shot unsupervised domain adaptive object detection are presented.

Low-Level Feature Alignment

Unlike the existing domain adaptation algorithms that typically align high-level features of CNN, in the proposed approach, the domain shift is reduced by aligning the low-level features. This idea is inspired by recent study of visualizing and understanding the feature maps of CNN [74]. This study shows that the upper layers of CNN respond to high-level features, such as the head of a dog, while the lower layers can be activated by low-level features, such as corners and edges. When a source domain image x_{S_i} and a target domain image x_T are inputted into a CNN, the output of the upper layer can be obtained as $f_H(x_{S_i})$ and $f_H(x_T)$. Since x_{S_i} and x_T may contain different objects, $f_H(x_{S_i})$ and $f_H(x_T)$ will be different. Thus, directly aligning the high-level features will not work. Nevertheless, x_{S_i} and x_T share common low-level features. Therefore, the output of the lower layer, $f_L(x_{S_i})$ and $f_L(x_T)$, will contain similar patterns. Thus, it is more reasonable to align the low-level features rather than the high-level features. In addition, aligning low-level features can benefit domain adaptive object detection because these features

are critical for detecting the shape of the object.

One-Shot Domain Adaptation

In the proposed OSFA, domain shift is reduced by aligning the average activation of feature maps in the lower layer with only one target sample. F is used to denote the activations of one feature map in the lower layer of a CNN. The activation of each unit in this feature map is denoted as $F_1, F_2, \dots, F_K$. It is assumed that $F_1, F_2, \dots, F_K$ are independent and identically distributed random variables whose mean and variance are μ and σ^2 , respectively. According to central limit theory, when K is large enough, the distribution of the average activation of this feature map is close to the normal distribution with mean μ and variance $\frac{\sigma^2}{K}$ as Eq. (8).

$$\bar{F} = \frac{\sum_{k=1}^{K=K} F_k}{K} \rightarrow \mathcal{N}\left(\mu, \frac{\sigma^2}{K}\right). \quad (8)$$

When $K \rightarrow \infty$, the variance $\frac{\sigma^2}{K} \rightarrow 0$. In this case, this distribution will become $\delta(\bar{F} - \mu)$ where $\delta(\cdot)$ denotes the Dirac delta function.

In the lower layer of a CNN, K is generally very large. For example, if the size of the input image is 512×512 , and VGG16 is used to extract features, the value of K in *Pool1* layer is 65536. Therefore, the distribution in Eq. (8) will approach the Dirac delta function, and it is reasonable to use only one sample to estimate the mean value μ of this distribution.

Architecture of the Proposed OSFA

The overall architecture of the proposed OSFA is shown in Fig. 6 [58]. The input contains two images: the labeled source image and the unlabeled target image. The labeled source image is inputted into the network on the top to calculate the detection loss, and the unlabeled target image is fed into the network on

the bottom. A domain loss is proposed to measure the domain shift between the source domain image and the target domain image. During the training process, the domain shift is reduced by minimizing this domain loss.

Figure 6: The overall architecture of the proposed OSFA.

The network on the top is the well-known object detection model SSD. In SSD, the backbone network is a truncated VGG-16 (from *Conv1_1* to *Pool5*). On top of this backbone network, some convolutional layers are stacked to generate the detection prediction at multiple scales. Specifically, the detection predictions are generated from the following convolutional layers: *Conv4_3*, *Conv7*, *Conv8_2*, *Conv9_2*, *Conv10_2*, *Conv11_2*, and *Conv12_2*. The details of SSD can be found in [75]. The bottom network has three layers, i.e., *Conv1_1*, *Conv1_2*, and *Pool1*. Their parameters are shared with the truncated VGG-16 on the top.

By feeding the source image x_{S_i} into the top network, the output of *Pool1* layer is $F_{S_i}^1$. Similarly, by inputting the target image x_T into the bottom network, the output of *Pool1* layer is F_T^1 . Then, the average activation of each feature map is calculated by applying global average pooling. Then, the domain loss L_{don} is

calculated as Eq. (9).

$$L_{don} = \frac{1}{B} \sum_{i=1}^B \sum_{j=1}^{M_1} |G_j(F_{S_i}^1) - G_j(F_T^1)|, \quad (9)$$

where $G_j(\cdot)$ represents the global average pooling operation on the j_{th} feature map; $|\cdot|$ calculates the element-wise absolute value; M_1 denotes the number of feature maps in *Pool1* layer; B represents the batch size.

Finally, the total loss is calculated as Eq. (10).

$$L = L_{det} + \alpha L_{don}, \quad (10)$$

where L_{det} is the detection loss that measures the difference between the groundtruth and detection predictions; α is a coefficient. During the training process, the whole model in Fig. 6 is used. During the evaluation process, only the top network is used.

2.2.3 Experimental Results

In this section, the effectiveness of the proposed OSFA is demonstrated in numerous experiments under two scenarios. The first scenario is to adapt from clear weather to foggy weather. In this scenario, domain shift arises from different weather condition. The capability to adapt between different weather conditions is crucial for the real-world data-driven decision-making. The second scenario is to adapt from synthetic images to real-world images. In this scenario, the source domain images are generated by a simulation engine, and the target domain images are collected in real-world. The simulation engine can easily produce a large number of synthetic images with computer-generated annotation. The ability to adapt from synthetic images to real-world images can avoid the burden of manually collecting and annotating real-world images.

Experimental Setup

The training data consist of the annotated source domain images and one unlabeled target domain image. The original SSD model is used as a baseline for comparison. For this baseline model, domain adaptation is not considered, and the model is trained with source domain data only. After the training process, the baseline model and the proposed OSFA are evaluated on the target domain data. In the following experiments, the mean average precision (mAP) is used as the evaluation metric, and the intersection over union (IOU) threshold is 0.5.

The hyperparameters for SSD are selected according to [75]. The value of the coefficient α is set as 0.001 for the following experiments. Stochastic gradient descent (SGD) with a momentum of 0.9 is used for the training. For the first 2k iterations, the learning rate is chosen as 5×10^{-4} to warm up the training. Then, the learning rate is set as 1×10^{-3} for the next 48k iterations. Then, for the next 10k iterations, the learning rate is reduced to 1×10^{-4} . For the final 10k iteration, the learning rate is further reduced to 1×10^{-5} .

Domain Adaptation from Clear Weather to Foggy Weather

In this section, the effectiveness of the proposed OSFA is demonstrated under the first scenario, i.e., adapting from clear weather to foggy weather. The source domain images are from *Cityscapes* dataset [76], which is collected in clear weather condition. The target domain images are from *Foggy Cityscapes* dataset [77], which is obtained by simulating fog on the images in *Cityscapes*. Both *Cityscapes* and *Foggy Cityscapes* are collected for semantic segmentation and do not directly provide bounding box annotations. According to the preprocessing step in [78], the bounding box annotation is obtained by taking the tightest rectangle of its instance. In *Cityscapes*, eight categories (person, rider, car, truck, bus, train, motorcycle, and bicycle) have instance labels. Thus, the experimental results are reported on

these categories. During the training process, *Cityscapes*'s training set is used as source domain data, and one unlabeled image is randomly selected from *Foggy Cityscapes*'s training set as the target domain data. After the training process, the baseline model and the proposed OSFA are evaluated on *Foggy Cityscapes*'s validation set.

The evaluation results of adapting from *Cityscapes* to *Foggy Cityscapes* are presented in Table 1. Without domain adaptation, the mAP of the baseline SSD model is 23.9%. The average precisions (APs) of the eight categories are 22.0%, 27.6 %, 37.0%, 16.7%, 28.2%, 9.9%, 19.8%, and 30.1%, respectively. Compared to this baseline model, the proposed OSFA with *Pool2* layer for feature alignment can improve the mAP to 27.2%. With *Pool1* layer for feature alignment, the mAP can be further improved to 28.7%, and the APs of the eight categories are 23.6%, 32.6%, 43.8%, 22.9%, 35.4%, 14.7%, 23.1%, and 33.2%, respectively. We can see that the proposed OSFA outperforms the baseline model on mAP and across all the eight categories.

Table 1: Adapt from *Cityscapes* to *Foggy Cityscapes*.

Methods	person	rider	car	truck	bus	train	motorcycle	bicycle	mAP
SSD [75]	22.0	27.6	37.0	16.7	28.2	9.9	19.8	30.1	23.9
img+ins align [78]	24.2	31.2	39.1	19.1	36.2	19.2	17.1	27.0	26.6
img+ins+cons [78]	25.0	31.0	40.5	22.1	35.3	20.2	20.0	27.1	27.6
OSFA (Pool2)	23.2	30.5	42.0	23.8	33.9	11.9	22.0	30.4	27.2
OSFA (Pool1)	23.6	32.6	43.8	22.9	35.4	14.7	23.1	33.2	28.7

The performance of the proposed OSFA is also compared with two state-of-the-art domain adaptation models in [78]. It is worth noting that these two models require the whole training set of *Foggy Cityscapes* for domain adaptation while the proposed OSFA only needs one image for domain adaptation. We can see that

the proposed OSFA with *Pool1* layer for feature alignment outperforms these two models. These results verify the effectiveness of the proposed OSFA in adapting from clear weather to foggy weather with only one unlabeled target image.

Domain Adaptation from Synthetic Images to Real-World Images

In this section, the effectiveness of the proposed OSFA is validated under the second scenario, i.e., adapting from synthetic images to real-world images. The source domain images are from *SIM 10k* dataset [79]. The images and the corresponding bounding box annotations in this dataset are generated by Grand Theft Auto V (GTA V), which is a simulation engine. In *SIM 10k*, only *car* category is labeled. Thus, the evaluation results is reported on this category. The target domain images are from three real-world datasets, i.e., *Cityscapes*, *PASCAL VOC2007*, and *PASCAL VOC2012*.

During the training process, *SIM 10k* is used as the source domain data, and one unlabeled image is randomly selected from *Cityscapes*'s training set as the target domain data. After the training process, the baseline and the proposed OSFA are evaluated on *Cityscapes*'s validation set.

The evaluation results of adapting from *SIM 10k* to *Cityscapes* are shown in Table 2. Without domain adaptation, the AP of the baseline SSD model is 35.1%. Compared to this baseline, the proposed OSFA with *Pool2* layer for feature alignment can improve the AP to 38.3%. Furthermore, with *Pool1* layer for feature alignment, the AP is improved to 38.6%.

The performance of the proposed OSFA is also compared with two state-of-the-art domain adaptation models in [78]. These two models require the whole training set of *Cityscapes* for domain adaptation while the proposed OSFA only needs one image for domain adaptation. We can observe that the performance of the proposed OSFA is close to the performance of these two models. These results

Table 2: Adapt from *SIM 10k* to *Cityscapes*.

Methods	car AP
SSD [75]	35.1
img+ins align [78]	37.86
img+ins+cons [78]	38.97
OSFA (Pool2)	38.3
OSFA (Pool1)	38.6

demonstrate the effectiveness of the proposed OSFA in adapting from synthetic images to real-world images with only one unlabeled target image.

To further evaluate the effectiveness of the proposed OSFA in adapting from synthetic images to real-world images, the following two experiments are conducted. In these experiments, *SIM 10k* is used as the source domain data, and one unlabeled image is randomly selected from the training set of *VOC2007* or *VOC2012* as the target domain data. After the training process, the baseline model and the proposed OSFA are evaluated on *VOC2007*'s test set and *VOC2012*'s validation set.

The evaluation results of adapting from *SIM 10k* to *PASCAL VOC* are presented in Table 3. Without domain adaptation, the APs of the baseline SSD model for these two experiments are 64.2% and 54.0%, respectively. In comparison with this baseline, the propose OSFA with *Pool2* layer for feature alignment can improve the APs to 69.1% and 57.5%, respectively. With *Pool1* layer for feature alignment, the APs for these two experiments can be further improved to 71.2% and 58.0%, respectively. These evaluation results demonstrate the proposed OSFA is effective in adapting to different target domains.

Table 3: Adapt from *SIM 10k* to *PASCAL VOC*.

Methods	VOC2007	VOC2012
SSD [75]	64.2	54.0
OSFA (Pool2)	69.1	57.5
OSFA (Pool1)	71.2	58.0

Ablation Experiments

To further understand the proposed OSFA, several ablation experiments are conducted. Specifically, this section first compares the performance of domain adaptation with different layers of CNN to show the advantage of the proposed low-level feature alignment. Then, this section verifies that the proposed OSFA can obtain good domain adaptation results regardless of the choice of the target sample. Finally, this section investigates the robustness of the proposed OSFA to the coefficient α .

The proposed OSFA uses lower layer of CNN, *Pool1* layer, for feature alignment. In this section, the performance of low-level feature alignment is compared with the performance of high-level feature alignment. The experimental setup and the training process are the same as those in Section “*Domain Adaptation from Clear Weather to Foggy Weather*”. Seven new models are trained with upper layers (*Conv4_3*, *Conv7*, *Conv8_2*, *Conv9_2*, *Conv10_2*, *Conv11_2*, and *Conv12_2*) for feature alignment, respectively. In these experiments, *Cityscapes* is used as the source domain data and *Foggy Cityscapes* is the target domain data.

The results of these seven models are shown in Table 4 and are compared with the baseline SSD model and the proposed OSFA with *Pool1* layer for feature alignment. The mAP of the baseline is 23.9%. The proposed OSFA with *Pool1* can improve the mAP to 28.7%. However, with the seven upper layers for feature

alignment, the mAPs are 25.2%, 24.6%, 25.5%, 25.0%, 25.8%, 25.6%, and 25.0%, respectively. These results are only slightly better than the result of the baseline model and are worse than the result of the proposed OSFA. From these experiments, we can see that low-level feature alignment is much better than high-level feature alignment.

Table 4: Feature alignment with different feature layers.

Methods	person	rider	car	truck	bus	train	motorcycle	bicycle	mAP
SSD [75]	22.0	27.6	37.0	16.7	28.2	9.9	19.8	30.1	23.9
Conv4_3	23.2	29.4	37.5	17.8	27.1	11.4	23.0	32.0	25.2
Conv7	23.6	31.6	38.5	15.0	28.2	9.1	18.2	32.2	24.6
Conv8_2	22.7	29.8	38.2	18.6	32.7	9.1	21.1	31.9	25.5
Conv9_2	22.7	30.7	38.3	18.7	29.7	9.7	19.6	30.8	25.0
Conv10_2	23.7	31.6	39.7	20.2	29.4	9.6	22.8	29.6	25.8
Conv11_2	24.2	31.9	38.5	19.0	29.6	9.3	20.3	31.9	25.6
Conv12_2	24.6	30.9	40.7	17.8	27.7	3.4	22.6	31.9	25.0
OSFA (Pool1)	23.6	32.6	43.8	22.9	35.4	14.7	23.1	33.2	28.7

In the proposed OSFA, an unlabeled target image is randomly selected for feature alignment. In the following experiments, the impact of the choice of the target sample on the domain adaptation performance will be analyzed. Four domain adaptation experiments are conducted:

- Adapt from Cityscapes to Foggy Cityscapes ($C \rightarrow F$)
- Adapt from SIM 10k to Cityscapes ($S \rightarrow C$)
- Adapt from SIM 10k to PASCAL VOC2007 ($S \rightarrow 07$)
- Adapt from SIM 10k to PASCAL VOC2012 ($S \rightarrow 12$)

Each experiment is run for five times. In each run, a different target sample is selected. The results for these four experiments are shown in Table 5. In $C \rightarrow F$, the mAP of the baseline is 23.9%. The mAPs of the five runs of the proposed OSFA are 28.7%, 29.9%, 28.0%, 28.5%, and 30.4%, respectively. The average value of these five runs is 29.1%. These results are all better than the result of the baseline. Similarly, in $S \rightarrow C$, the mAP of the baseline is only 35.1%. The mAPs of the five runs of the proposed OSFA are 38.6%, 38.5%, 38.6%, 37.7%, and 38.3%, respectively. The average value of these five runs is 38.3%. In $S \rightarrow 07$, the mAP of the baseline is 64.2%. The mAPs of the five runs of the proposed OSFA are 71.2%, 71.0%, 70.4%, 71.5%, and 70.3%, respectively. The average value of these five runs is 70.9%. In $S \rightarrow 12$, the mAP of the baseline is 54.0%. The mAPs of the five runs of the proposed OSFA are 58.0%, 57.5%, 58.2%, 57.2%, and 58.1%, respectively. The average value of these five runs is 57.8%. These results show that the proposed OSFA can obtain good domain adaptation results regardless of the choice of the target sample.

Table 5: Domain adaptation with different target sample.

Methods	$C \rightarrow F$	$S \rightarrow C$	$S \rightarrow 07$	$S \rightarrow 12$
SSD	23.9	35.1	64.2	54.0
OSFA (run1)	28.7	38.6	71.2	58.0
OSFA (run2)	29.9	38.5	71.0	57.5
OSFA (run3)	28.0	38.6	70.4	58.2
OSFA (run4)	28.5	37.7	71.5	57.2
OSFA (run5)	30.4	38.3	70.3	58.1
OSFA (average)	29.1	38.3	70.9	57.8

In the following experiments, the robustness of the proposed OSFA to the coefficient α is investigated. The experimental setup and the training process are the same as those in Section “*Domain Adaptation from Clear Weather to Foggy Weather*”. Six OSFA models are trained with different coefficient $\alpha = \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}$. The experimental results are shown in Table 6. With these different coefficients α , the corresponding mAPs are 27.6%, 28.8%, 28.7 %, 29.4%, 28.7%, and 28.5%, respectively. These results are all better than the baseline model without domain adaptation. In addition, when the coefficient α varies from 0.0001 to 0.05, the mAPs are around 28.0%. These results show that the proposed OSFA is robust to the choice of coefficient α .

Table 6: Domain adaptation with different coefficient α .

Methods	person	rider	car	truck	bus	train	motorcycle	bicycle	mAP
SSD [75]	22.0	27.6	37.0	16.7	28.2	9.9	19.8	30.1	23.9
OSFA ($\alpha = 0.0001$)	23.2	30.7	41.9	22.4	32.7	13.8	24.3	31.9	27.6
OSFA ($\alpha = 0.0005$)	24.2	32.4	43.6	22.7	35.8	16.2	24.2	31.3	28.8
OSFA ($\alpha = 0.001$)	23.6	32.6	43.8	22.9	35.4	14.7	23.1	33.2	28.7
OSFA ($\alpha = 0.005$)	22.5	31.8	45.0	19.7	36.6	20.1	25.7	33.7	29.4
OSFA ($\alpha = 0.01$)	22.2	32.6	43.2	24.9	32.7	19.6	23.0	31.4	28.7
OSFA ($\alpha = 0.05$)	22.5	32.3	44.5	18.7	35.3	23.0	21.7	30.4	28.5

Insights and Discussions

In this section, experiments are conducted to show that the proposed OSFA succeeds in reducing domain shift. The process is the same as that in Section “*Domain Adaptation from Clear Weather to Foggy Weather*”. A baseline SSD model without domain adaptation and the proposed OSFA with *Pool1* layer for feature alignment are trained. Specifically, the baseline model is trained only with

Cityscapes dataset. For the proposed OSFA, *Cityscapes* is used as the source domain data, and one unlabeled image is randomly selected from *Foggy Cityscapes* as the target domain data. After the training process, the domain shift for the baseline model and the proposed OSFA is calculated. The input for each model is a pair of images: x_{S_i} and x_{T_i} . x_{S_i} is an image from *Cityscapes* while x_{T_i} is from *Foggy Cityscapes* and is generated by simulating fog on x_{S_i} . Therefore, x_{S_i} and x_{T_i} will have the same objects. The only difference between them is that there is fog in x_{T_i} . This fog is the source of domain shift. This experiment will validate whether the proposed OSFA can reduce this type of domain shift.

When x_{S_i} and x_{T_i} are inputted into the model, the output of *Pool1* layer can be obtained as $p_{S_i}^1$ and $p_{T_i}^1$. Then, the domain shift in *Pool1* layer is calculated as Eq. (11).

$$D_{p^1} = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^M G_j \left(\left| \frac{p_{S_i}^1 - p_{T_i}^1}{p_{S_i}^1} \right| \right), \quad (11)$$

where $|\cdot|$ calculates the element-wise absolute value; $G_j(\cdot)$ represents the global average pooling operation over the j_{th} feature map; M is the number of feature maps in *Pool1* layer; N represents the number of samples in *Cityscapes*'s training set. Then, the domain shift reduction at *Pool1* layer is calculated as $(D_{p^1}^{base} - D_{p^1}^{OSFA}) / D_{p^1}^{base}$, where $D_{p^1}^{OSFA}$ represents the domain shift of the proposed OSFA; $D_{p^1}^{base}$ is the domain shift of the baseline model. The domain shift reduction for the other eight layers, including *Pool2*, *Conv4_3*, *Conv7*, *Conv8_2*, *Conv9_2*, *Conv10_2*, *Conv11_2*, *Conv12_2*, can be calculated in a similar way.

The domain shift reduction for these nine layers are 67.39%, 18.39%, 14.85%, 26.77%, 17.61%, 2.90%, 0.81%, 14.58%, and 17.03%, respectively. Since *Pool1* layer is used in the proposed OSFA for feature alignment, OSFA significantly reduces the domain shift at *Pool1* layer by 67.39%. For the remaining eight layers,

even though they are not used for feature alignment, OSFA can still reduce the domain shift at these layers. These results demonstrate that the proposed OSFA is effective in reducing domain shift.

2.3 Unsupervised Feature Extraction

Unlike the supervised feature extractors that require labeling information and suffer from domain shift, unsupervised feature extractors can automatically learn underlying patterns from the input data without labeling information and therefore does not have the domain shift problem. In this section, a widely used unsupervised feature extractor, autoencoder [80], will be introduced.

2.3.1 Autoencoder

The architecture of autoencoder is shown in Fig. 7 [80]. It contains two parts: encoder and decoder. The encoder encodes input data into hidden layer while the decoder decodes information from hidden layer to output layer. The output of hidden units can be treated as features of input data.

Figure 7: Architecture of autoencoder.

The output of hidden units is shown in Eq.(12) where W' is the weight matrix,

and b' is the bias vector. In Eq.(12), $x^{(j)}$ denotes a sample of input data, and $a^{(j)}$ denotes the output of hidden units. The activation function $f(.)$ is the sigmoid function.

$$a^{(j)} = f(W'x^{(j)} + b') \quad (12)$$

The output of autoencoder is shown in Eq.(13) where W is the weight matrix, and b is the bias vector. In Eq.(13), $\hat{x}^{(j)}$ denotes the output of autoencoder when input data $x^{(j)}$ is fed into autoencoder.

$$\hat{x}^{(j)} = f(Wa^{(j)} + b) \quad (13)$$

Autoencoder is trained to minimize the error between input data and the output of autoencoder. Therefore, the cost function of autoencoder can be expressed as Eq.(14) where m denotes the number of samples in input data, $x^{(j)}$ denotes a sample of input data, and $a^{(j)}$ denotes the output of hidden units when given the input $x^{(j)}$. The optimal weight matrix and bias vector can be obtained by minimizing the cost function.

$$J(W, b, W', b') = \frac{1}{2m} \sum_{j=1}^m [x^{(j)} - f(Wa^{(j)} + b)]^2 \quad (14)$$

In order to extract structured features, a sparse penalty can be imposed on the hidden units of autoencoder [81, 82]. Introducing sparsity is an efficient way to represent information with a small number of activated hidden units [83, 84]. This is inspired by the observation that natural images can be represented by sparse features [85]. When natural images are filtered with log-Gabor filters, the output distributions typically show sparse structures [86].

Sparse autoencoder has been widely used in automatic feature extraction. For instance, Poultney *et al.* [87] use sparse autoencoder to extract stroke-like

features from handwriting digits dataset. Le *et al.* [88] propose a nine-layer locally connected sparse autoencoder to learn face features from unlabeled data. These features are robust to translation and scaling. Chen *et al.* [89] use a stacked sparse autoencoder to identify and classify overvoltage waveforms in power distribution systems. Sparse autoencoder can automatically extract discriminative features from the ferroresonance overvoltage waveforms. Tao *et al.* [90] propose a stacked sparse autoencoder to learn spectral-spatial features from hyperspectral imagery which is intrinsically defined in spectral and spatial domains.

Although sparse autoencoder is effective in feature extraction, a sparse hyperparameter should be determined to balance the trade-off between the reconstruction error and the sparsity of sparse autoencoder. Many researchers have explored how to determine the sparse hyperparameter. For example, Chen *et al.* [89] and Sun *et al.* [91] search for the sparse hyperparameter within a large interval. Unlike these searching methods, Gong *et al.* [92] introduce a multiobjective evolutionary algorithm to determine the sparse hyperparameter. The evolutionary algorithm obtains a set of Pareto optimal solutions and selects a solution from the knee areas on the Pareto optimal front. Although the above methods can choose a suitable sparse hyperparameter, they are time-consuming, especially when the dataset is large. Besides, they do not show the relationship between the sparse hyperparameter and the sparsity.

2.3.2 Generative Model for Sparse Autoencoder

In this section, in order to efficiently determine the sparse hyperparameter, a generative model is derived for sparse autoencoder from the probabilistic perspective [59].

According to [93], the input data x can be represented by the output of sparse autoencoder $\hat{x}$ and additive Gaussian white noise v as

$$\begin{aligned}
x &= \hat{x} + v \\
&= f(Wa + b) + v
\end{aligned} \tag{15}$$

where a is the output of hidden units. Eq.(15) can be rewritten from a probabilistic perspective as Eq.(16).

$$\begin{aligned}
P(x | a, W, b) &= \mathcal{N}(x | f(a, W, b), \sigma) \\
&= \frac{1}{Z} \exp \left[-\frac{(x - f(Wa + b))^2}{2\sigma^2} \right]
\end{aligned} \tag{16}$$

where Z is a normalizing constant, and σ^2 denotes the variance of the Gaussian white noise.

In order to obtain a generative model for sparse autoencoder, the conditional distribution $P(x | W, b)$ is derived as Eq.(17).

$$P(x | W, b) = \int P(x | a, W, b) P(a) d(a) \tag{17}$$

where $P(a)$ denotes the prior distribution of hidden units. Since the hidden units are independent, the prior distribution can be rewritten as Eq.(18) where k is the number of hidden units.

$$P(a) = \prod_{i=1}^k P(a_i) \tag{18}$$

When the sparse penalty is imposed on the hidden units, the output of hidden units should be near zero with high probability. Therefore, the average activation of hidden unit i defined in Eq.(19) should be a small value near zero. In Eq.(19), m denotes the number of samples in input data, and $a_i^{(j)}$ denotes the output of

hidden unit i .

$$\rho_i = \frac{1}{m} \sum_{j=1}^m a_i^{(j)} \quad (19)$$

From the probabilistic perspective, when the sparse penalty is imposed, the prior distribution $P(a_i)$ should have a peak at zero and have low probability anywhere else. A suitable form of this prior distribution is introduced by Olshausen *et al.* [94] and shown in Eq.(20)

$$P(a_i) = \frac{1}{Z_\beta} \exp [-\beta S(a_i)] \quad (20)$$

where the coefficient β determines the shape of the prior distribution, $S(a_i)$ corresponds to the sparse penalty, and Z_β is a normalizing constant.

In the generative model, L2 norm is used as sparse penalty. By substituting L2 norm sparse penalty $S(a_i) = a_i^2$ into Eq.(20), the prior distribution $P(a_i)$ can be obtained as Eq.(21) where the normalizing constant Z_β equals to $\sqrt{\pi/4\beta}$.

$$P(a_i) = \frac{1}{Z_\beta} \exp (-\beta a_i^2) \quad (21)$$

From the probabilistic perspective, sparse autoencoder is trained to make the conditional distribution $P(x | W, b)$ close to the input data distribution $P^*(x)$ by choosing suitable weight matrix W and bias vector b . In order to make these two distributions close, the KL divergence between distribution $P(x | W, b)$ and $P^*(x)$ is minimized. The KL divergence is shown in Eq.(22).

$$\begin{aligned} & D(P^*(x) \parallel P(x | W, b)) \\ &= \int P^*(x) \log \left[\frac{P^*(x)}{P(x | W, b)} \right] dx \\ &= \int P^*(x) \log [P^*(x)] dx - \int P^*(x) \log [P(x | W, b)] dx \end{aligned} \quad (22)$$

Since distribution $P^*(x)$ is constant across our choice of weight matrix W and bias vector b , the first term in Eq.(22) is constant. Therefore, minimizing the KL divergence is equivalent to maximize $\langle \log [P(x | W, b)] \rangle$ where $\langle . \rangle$ denotes the expectation over input data.

In order to minimize the KL divergence, the distribution $P(x | W, b)$ should be estimated. However, in Eq.(17), the integral over a to obtain $P(x | W, b)$ is generally intractable. Fortunately, due to the sparse assumption of distribution $P(a)$, the shape of distribution $P(a)$ has a peak at zero. Therefore, distribution $P(x, a | W, b) = P(x | a, W, b)P(a)$ has a fairly tightly peaked maximum. This integral can be approximated by the maximum value of $P(x, a | W, b)$ [94]. Then, an approximated solution is shown in Eq.(23).

$$\left(W, b, W', b' \right)^* = \operatorname{argmax}_{W, b, W', b'} \langle \max_a \log [P(x | a, W, b)P(a)] \rangle \quad (23)$$

The next step is to substitute the conditional distribution $P(x | a, W, b)$ and prior distribution $P(a)$ into $\log [P(x | a, W, b)P(a)]$ to get Eq.(24).

$$\begin{aligned} & \log [P(x | a, W, b)P(a)] \\ &= \log \left\{ \frac{1}{Z} \exp \left[-\frac{[x - f(Wa + b)]^2}{2\sigma^2} \right] \right\} \\ &+ \log \left[\prod_{i=1}^k \frac{1}{Z_\beta} \exp(-\beta a_i^2) \right] \\ &= \log \frac{1}{Z} - \frac{[x - f(Wa + b)]^2}{2\sigma^2} + \log \frac{1}{Z_\beta^k} - \sum_{i=1}^k \beta a_i^2 \\ &= \frac{-1}{2\sigma^2} \left\{ [x - f(Wa + b)]^2 + 2\sigma^2 \beta \sum_{i=1}^k a_i^2 \right\} - \log Z_\beta^k Z \end{aligned} \quad (24)$$

In Eq.(24), the second term $\log Z_\beta^k Z$ is a constant. In the first term, $[x - f(Wa + b)]^2$ is the reconstruction error, and $2\sigma^2 \beta \sum_{i=1}^k a_i^2$ is the sparse

penalty term. Therefore, the sparse hyperparameter λ equals to $2\sigma^2\beta$ where σ^2 is the variance of the reconstruction error, and β is the coefficient determining the shape of the prior distribution. Eq.(23) can be rewritten in the form of the cost function as Eq.(25) where m denotes the number of samples in input data, and $a^{(j)}$ is the output of hidden units. Maximizing Eq.(23) is equivalent to minimize this cost function.

$$J(W, b, W', b') = \frac{1}{2m} \sum_{j=1}^m [x^{(j)} - f(Wa^{(j)} + b)]^2 + \frac{1}{2m} \lambda \sum_{j=1}^m \sum_{i=1}^k [a_i^{(j)}]^2 \quad (25)$$

In Eq.(25), the first term is the reconstruction error, and the second term is the sparse penalty term. The sparse hyperparameter λ should be determined to balance the trade-off between the reconstruction error and the sparse penalty term.

In order to determine the sparse hyperparameter, first of all, the relationship between the sparse hyperparameter and the average activation of hidden units is derived. This relationship is shown in Eq.(26)

$$E(a_i) = \rho_i = \frac{1}{\sqrt{\pi\beta}} \quad (26)$$

where β is the coefficient determining the shape of the prior distribution. The average activation ρ_i equals to the expectation of the sparse prior distribution $P(a_i)$. Therefore, Eq.(26) can be used to estimate the average activation of hidden units. In addition, it provides an explicit guidance to determine the sparse hyperparameter.

The relationship between the average activation and the coefficient β is shown in Fig. 8. When β is small, the average activation is high, and hidden units become

less sparse. When β is large, good sparsity can be obtained. However, the sparse hyperparameter λ , which equals to $2\sigma^2\beta$, will become large. When we minimize the cost function in Eq.(25), large λ will result in large reconstruction error.

Figure 8: Relationship between the average activation and the coefficient β .

Inspired by multi-objective optimization[95], in Fig. 8, a knee region is defined from 8 to 16. When β is larger than 16, increasing β will benefit sparsity little. When β is smaller than 8, decreasing β will worsen sparsity a lot. The knee region is defined to make β as small as possible while obtaining good sparsity. Choosing β inside the knee region balances the trade-off between the reconstruction error and the sparsity. However, when β is chosen outside the knee region, it will result in either large reconstruction error or high average activation.

In order to determine the sparse hyperparameter λ , which equals to $2\sigma^2\beta$, β and σ^2 should be determined. In Fig. 8, β is chosen from the center of the knee region to balance the trade-off between the reconstruction error and the sparsity. The variance σ^2 can be estimated during the training process. Specifically,

experimental results show that, when β is chosen from the knee region, the standard deviation of sparse autoencoder increases about 20% in comparison with the standard deviation of autoencoder without sparse penalty. Therefore, an autoencoder without sparse penalty is trained to estimate the standard deviation of reconstruction error σ_1 . Then, the sparse hyperparameter can be determined by $\lambda = 2 \times (1.2\sigma_1)^2 \beta$.

2.3.3 Experimental Results

This section will first evaluate the validity of the proposed approach in estimating the average activation of hidden units. Then, the validity of the proposed sparse hyperparameter determination method will be evaluated on MNIST dataset [40].

Average activation estimation

The effectiveness of the proposed approach in estimating the average activation of hidden units is verified on MNIST dataset. The 28×28 input image is first flattened into a vector. This vector will be fed into the sparse autoencoder, which has 784 input units. The number of hidden units is selected as 169. The sparse hyperparameter λ varies from 0.1 to 0.5 with step 0.1. After the training process, the average activation of hidden units is calculated. This experimental average activation is compared with the average activation calculated by Eq.(26) where β equals to $\lambda/2\sigma^2$.

The experimental result is shown in Table 7. When λ increases from 0.1 to 0.5, the average activation of hidden units decreases. Besides, the experimental average activation matches the average activation calculated by Eq.(26) very well. This experimental result demonstrates that Eq.(26) can effectively estimate the average activation of hidden units.

Table 7: Average activation of hidden units on MNIST dataset

λ	Experimental activation	β	Activation calculated by Eq.(26)
0.1	0.2893	3.90	0.2859
0.2	0.2204	7.09	0.2120
0.3	0.1850	9.83	0.1800
0.4	0.1621	12.28	0.1610
0.5	0.1432	14.50	0.1482

Sparse hyperparameter determination

The effectiveness of the proposed method in determining the sparse hyperparameter is validated on MNIST dataset. MNIST dataset is divided into two parts. Part I contains digits from 5 to 9 while Part II contains digits from 0 to 4. Part I is used to train the sparse autoencoder to learn to extract features. It is worth noting that Part I does not contain any labeling information, which means that the training process is unsupervised. After this training process, the weight matrix and bias vector of the sparse autoencoder are fixed. This trained sparse autoencoder will be used as feature extractor. Then, the performance of the sparse autoencoder in feature extraction is evaluated on Part II. Specifically, the sparse autoencoder is used to extract features from the samples in Part II. Then, these features are fed into a classifier. The samples in Part II are further divided into training set and test set. The classifier is trained with the training set and evaluated on the test set. The accuracy on the test set is used as the metric to evaluate the performance of the sparse autoencoder in feature extracting. In other words, if the sparse autoencoder can extract good features from the input data, it will result in high accuracy. The size of the input image is 28×28 , so the sparse autoencoder has 784 input units. The number of hidden units is selected as 169.

In order to determine the sparse hyperparameter, first of all, an autoencoder without sparse penalty is trained to estimate the standard deviation of reconstruction error σ_1 , which equals to 0.1083. Then, $\beta = 12$ is chosen from the center of the knee region in Fig. 8. Therefore, the sparse hyperparameter chosen by the proposed method equals to $2 \times (1.2 \times \sigma_1)^2 \times \beta = 0.4054$. In order to evaluate the validity of the proposed method, some β are chosen outside the knee region. Specifically, $\beta_1 = 8$ and $\beta_2 = 16$ are chosen from the edge of the knee region. Their corresponding sparse hyperparameters are $\lambda_1 = 0.2702$ and $\lambda_2 = 0.5405$. Besides, we choose $\beta_3 = 0$, $\beta_4 = 4$, and $\beta_5 = 20$, which are outside the knee region. Their corresponding sparse hyperparameters are $\lambda_3 = 0$, $\lambda_4 = 0.1351$, and $\lambda_5 = 0.6756$. Then, the sparse autoencoder is trained with these sparse hyperparameters to extract features, respectively. Finally, these features are fed into the classifier.

Experimental results are shown in Table 8. The classification accuracy of sparse autoencoder is better than that of autoencoder without sparse penalty. When β increases from 0 to 12, the classification accuracy increases from 97.26% to 98.01%. When β continues to increase from 12 to 20, the classification accuracy decreases from 98.01% to 97.88%. In Table 8, $\beta = 12$ chosen by the proposed method achieves the best accuracy. When $\beta = 8$ and $\beta = 16$ are chosen from the edge of the knee region, the classification accuracy is close to the best accuracy. This result demonstrates that the optimal β should be in the knee region, and the optimal accuracy is close to 98.01%. When β is chosen from the knee region, the sparse autoencoder can achieve good classification accuracy. However, when β is chosen outside the knee region, the classification accuracy becomes worse.

Table 8 also shows that, when $\beta = 12$ is chosen from the knee region, the standard deviation increase about 20% in comparison with the standard deviation when $\beta = 0$.

Table 8: Classification accuracy of sparse autoencoder with 169 hidden units.

β	λ	σ	Classification accuracy
0	0	0.1083	97.26%
4	0.1351	0.1147	97.79%
8	0.2702	0.1224	97.96%
12	0.4054	0.1279	98.01%
16	0.5405	0.1329	98.00%
20	0.6756	0.1378	97.88%

Traditional sparse hyperparameter determination method [89] searches for a suitable sparse hyperparameter within a large range. For example, in this experiment, the sparse hyperparameter λ may be chosen from 0.1 to 1 with step 0.1. Therefore, the traditional method has to train 10 sparse autoencoders and compare their classification performance. Then, the traditional method can find a suitable sparse hyperparameter 0.4, but this searching process is time-consuming. Moreover, in general, it is hard to determine the range and the step when choosing the sparse hyperparameter. Therefore, it will take a long time to choose a suitable sparse hyperparameter. The advantage of the proposed method is that it can directly determine a suitable sparse hyperparameter.

To demonstrate the effectiveness of the proposed method, features learned by sparse autoencoder with different sparse hyperparameters are shown in Fig. 9. In Fig. 9a, autoencoder without sparse penalty learns unstructured features. When the sparse constraint is imposed on autoencoder, sparse autoencoder can learn structured features. However, when sparse hyperparameter is large, many features become similar. Fig. 9d shows the features learned by the sparse autoencoder whose sparse hyperparameter is determined by the proposed method. These

(a) $\lambda = 0$

(b) $\lambda = 0.1351$

(c) $\lambda = 0.2702$

(d) $\lambda = 0.4054$

(e) $\lambda = 0.5405$

(f) $\lambda = 0.6756$

Figure 9: Features learned by sparse autoencoder with 169 hidden units.

features are structured and have few similar features.

Fig. 10 shows the activation of hidden units when a sample from MNIST

(a) Autoencoder ($\lambda = 0$)

(b) Sparse autoencoder ($\lambda = 0.4054$)

Figure 10: Activation of hidden units in autoencoder and sparse autoencoder.

dataset is fed into autoencoder and sparse autoencoder. Fig. 10a shows the activation of autoencoder hidden units. Most of these hidden units are active. In Fig. 10b, when sparse penalty is introduced, most of the hidden units become inactive. These results show that sparse autoencoder is effective in extracting sparse features.

To further evaluate the validity of the proposed sparse hyperparameter determination, I conduct the following experiments for sparse autoencoder with 196

hidden units. First, an autoencoder is trained without sparse penalty. The standard deviation of the reconstruction error σ_1 is 0.1058. Then, $\beta = 12$ is chosen from the center of the knee region in Fig. 8. Therefore, the sparse hyperparameter chosen by the proposed method equals to $2 \times (1.2 \times \sigma_1)^2 \times \beta = 0.3869$. I also choose some β outside the knee region as benchmark. Specifically, $\beta_1 = 8$ and $\beta_2 = 16$ are chosen from the edge of the knee region. Their corresponding sparse hyperparameters are $\lambda_1 = 0.2579$ and $\lambda_2 = 0.5158$. I also choose $\beta_3 = 0$, $\beta_4 = 4$, and $\beta_5 = 20$, which are outside the knee region. Their corresponding sparse hyperparameters are $\lambda_3 = 0$, $\lambda_4 = 0.1290$, and $\lambda_5 = 0.6448$. Then, the sparse autoencoders with these sparse hyperparameters are trained to extract features. Finally, these features are fed into the classifier.

The experimental results are shown in Table 9. We can see that the classification accuracy of sparse autoencoders is better than that of autoencoder without sparse penalty. When β increases from 0 to 12, the classification accuracy increases from 97.30% to 98.16%. When β continues to increase from 12 to 20, the classification accuracy decreases from 98.16% to 97.92%. In Table 9, $\beta = 12$ chosen by the proposed method achieves the best accuracy. When $\beta = 8$ and $\beta = 16$ are chosen from the edge of the knee region, the classification accuracy is close to the best accuracy. This result demonstrates that the optimal β should be in the knee region, and the optimal accuracy is close to 98.16%. When β is chosen from the knee region, the sparse autoencoder can achieve good classification accuracy. However, when β is chosen outside the knee region, the classification accuracy becomes worse. Table 9 also shows that, when $\beta = 12$ is chosen from the knee region, the standard deviation increase about 20% in comparison with the standard deviation when $\beta = 0$.

Features learned by sparse autoencoder with different sparse hyperparameters

Table 9: Classification accuracy of sparse autoencoder with 196 hidden units.

β	λ	σ	Classification accuracy
0	0	0.1058	97.30%
4	0.1290	0.1128	97.90%
8	0.2579	0.1207	98.09%
12	0.3869	0.1265	98.16%
16	0.5158	0.1318	98.06%
20	0.6448	0.1364	97.92%

are shown in Fig. 11. In Fig. 11a, autoencoder without sparse penalty learns unstructured features. When the sparse constraint is imposed on autoencoder, sparse autoencoder can learn structured features. However, when sparse hyperparameter is large, many features become similar. Fig. 11d shows the features learned by the sparse autoencoder whose sparse hyperparameter is determined by the proposed method. These features are structured and have few similar features.

2.4 Chapter Summary

This chapter has introduced supervised feature extractor for automatic feature extraction, which is a crucial part for data-driven decision-making. The domain shift problem for supervised feature extraction has been investigated. The experimental results have shown that the performance of the supervised feature extractor will deteriorate sharply because of the domain shift. To reduce the impact of domain shift, a domain adaptation method called OSFA has been proposed to align the low-level features of the source domain and the target domain. The experimental results have shown that with the proposed OSFA, the feature extractor can be adapted between different domains. The experimental results on adapting from

(a) $\lambda = 0$

(b) $\lambda = 0.1290$

(c) $\lambda = 0.2579$

(d) $\lambda = 0.3869$

(e) $\lambda = 0.5158$

(f) $\lambda = 0.6448$

Figure 11: Features learned by sparse autoencoder with 196 hidden units.

clear weather to foggy weather and adapting from synthetic images to real-world images have demonstrated the effectiveness of the proposed OSFA. The ablation

results have shown that compared to aligning high-level features, the proposed low-level feature alignment is much more effective in domain shift reduction. In addition, the ablation results have shown that the proposed OSFA can obtain good domain adaptation results regardless of the choice of the target sample.

This chapter has also introduced unsupervised feature extractor, which does not require any labeling information and therefore does not suffer from the domain shift problem. A widely used unsupervised feature extractor, sparse autoencoder, has been introduced. The sparse hyperparameter determination problem has been investigated in this chapter. Unlike the previous works that use brute-force search to select this hyperparameter, a generative model has been established to efficiently determine the sparse hyperparameter. In addition, the generative model can be used to analyze the relationship between the sparse hyperparameter and the sparsity of the hidden units of sparse autoencoder. The experimental results on MNIST dataset have shown that the proposed method is effective in estimating the average activation of hidden units and determining the sparse hyperparameter. With the proposed sparse hyperparameter determination method, the sparse autoencoder can extract structured features.

CHAPTER 3

Heterogeneous Data Fusion for Decision-Making

3.1 Chapter Overview

Real-world decision-making generally needs to understand and process data from different sources. These heterogeneous data have different types and formats. Therefore, after automatic feature extraction, the next important step is to properly fuse these data to support the decision-making process. Heterogeneous data fusion is a crucial part for data-driven decision-making. It generally contains three major steps. The first step is to refine the features according to their correlation. In other words, this step will keep the highly correlated features and discard the irrelevant features. Then, these features should be properly normalized and transformed from different sources into a common feature space. Finally, suitable fusion operation should be designed for effective feature fusion.

This chapter will introduce how to fuse image and natural language. Image is a kind of spatial-correlated data while natural language is a kind of time-series data. These two types of data are commonly used in data-driven decision-making. In addition, many other types of data can be processed either as spatial-correlated data or as time-series data. Therefore, it is of great importance to develop fusion techniques for these two types of data. The developed fusion techniques can also benefit the fusion of other types of data. The performance of the data fusion technique is validated in a data-driven decision-making task, visual question answering (VQA). The main contributions are twofold. First, an attention network is used to capture the correlation between different types of data and refine the input data according to the correlation. Second, a hierarchical fusion network is developed to transform the input data into a common feature space and effectively fuse them.

3.2 Background and Related Works

In recent years, image and natural language fusion has attracted wide attention from researchers. For example, inspired by image captioning model, Neural-Image-QA [96] and VIS+LSTM [97] are proposed for image and natural language fusion. In these works, image features extracted by CNN are treated as the starting word of the question. Then, the image features and the question are fused in an LSTM network. In a similar work, the ACK model in [98] is proposed in similar to the VIS+LSTM model, except that it uses image captions, image attributes, and relevant external knowledge as the input to the LSTM network. Although these models provide some new ideas for image and natural language fusion, LSTM network has very limited fusion capability.

To effectively fuse image features with question features, researchers incorporate the spatial attention mechanism [99, 100, 101]. For instance, Chen *et al* [99] propose a configurable convolution operation to find which part of the image is related to the question. The kernels of this convolution operation are determined by the input question, which means this model needs to generate different kernels for different questions. Xu *et al* [100] further propose a spatial memory network for heterogeneous data fusion. This model uses CNN to extract image features and bag-of-words model to extract question features. The spatial attention matrix is calculated by dot product of the image features and the question features. It can automatically determine which image region is related to the input question. This model achieves promising performance, however, one limitation is that the bag-of-words model cannot capture the semantic representations of the question. To overcome this limitation, the SAN model in [101] uses LSTM network to extract these semantic representations. The spatial attention matrix is calculated by adding the image features with the question features. This attention process is

repeated twice to improve the performance.

Inspired by the recent progresses in natural language processing, Andreas *et al* [102] decomposed each question into their linguistic substructures which are processed by module networks. These module networks determine which image regions are related to the input question. Similar to [102], the D-NMN model in [103] uses module network to process the linguistic substructure. In addition, a learned syntactic analysis is used to assemble a collection of module networks into a deep neural network to generate the answer. While these approaches have shown promising results, one limitation is that they require hand-crafted module networks, which are unsuitable to handle a large number of questions in real-world scenario.

The effectiveness of models in fusing image and natural language is verified in a data-driven decision-making task, VQA. For VQA task, the input consists of an image V and a related question q . The optimal answer a^* is generated based on Eq. (27)

$$a^* = \underset{a \in \Omega}{\operatorname{argmax}} p(a \mid V, q; \theta) \quad (27)$$

where Ω is a set of possible answers, and θ is the parameters of the model. The model needs to generate an answer to the question by effectively fusing the image and the question. An example of the VQA task is shown in Fig. 12 [104]. In order to correctly answer this question, the model needs to fuse the image and question to recognize people and detect the number of people in the image. As a task involving heterogeneous data, in general, VQA task is much more challenging than those tasks only involving single type of data.

Question: How many people are in the photo?
Answer: 2

Figure 12: An example for the VQA task.

3.3 Feature Fusion for Heterogeneous Data

To effectively fuse image and natural language and support the decision-making process in VQA task, a deep neural network-based model called AnswerNet is proposed [60]. The overall architecture of the proposed AnswerNet is shown in Fig. 13. It has three major parts: feature extraction, feature fusion, and answer generation.

Figure 13: Overall architecture of the proposed AnswerNet.

Before feature fusion, the first step is to extract high-level features from the

image and question. In AnswerNet, discriminative image features are extracted by a state-of-the-art CNN, Deep Residual Net [45]. Before feeding the image into the Deep Residual Net, it is resized into 224×224 . The output of the Deep Residual Net has 2048 feature maps whose dimension is 7×7 . These features are normalized by L2 normalization as

$$V^* = \frac{V}{\sqrt{\max(\sum_{i=1}^7 \sum_{j=1}^7 \sum_{k=1}^{2048} V_{i,j,k}^2, \epsilon)}} \quad (28)$$

where V denotes the images features extracted by Deep Residual Net; V^* represents the normalized image features; ϵ is a small value to avoid dividing by zero. For the question features, the semantic information of the question is extracted by LSTM network, and the term frequencies of distinct words are captured by bag-of-words model. After feature fusion, the fused features will be fed into a softmax layer to predict the answer from a set of answer candidates.

Actually, feature fusion is the most important part for this data-driven decision-making task. It contains two networks: attention network and fusion network.

3.3.1 Attention Network

Since the image contains rich information and many different objects, attention network is designed to refine the image features according to the question. Specifically, the attention network is used to keep the image features that are relevant to the question and discard the irrelevant features. The architecture of the attention network is shown in Fig. 14.

In Fig. 14, if we divide the input image into 7×7 grid, then each slice of the image will correspond to a unit on the feature map. The goal of the attention network is to determine which slices are relevant to the question. In the attention network, first, the question features are tiled to match the dimension of the image

Figure 14: Architecture of the attention network.

features. The correlation between the question and each slice of the image is calculated by Hadamard product, which is element-wise multiplication operation for tensors. In the attention network, two convolutional layers are first used to extract the spatial relationship from different units on the feature map. For the first convolutional layer, the kernel size is 1, and the number of kernels is 512. For the second convolutional layer, the kernel size is 1, and the number of kernels is 1. Then, a softmax layer is used to generate the attention weights as Eq. (29).

$$W_{i,j} = \frac{\exp(S_{i,j})}{\sum_{j=1}^7 \sum_{i=1}^7 \exp(S_{i,j})} \quad (29)$$

where $S_{i,j}$ denotes the unit on the output of the second convolutional layer at position (i, j) ; $W_{i,j}$ represents the attention weight. The dimension of the attention weights is also 7×7 . Each weight corresponds to a slice in the input image. Large weight means that this slice is relevant to the question.

With these attention weights, soft-attention is used to get the refined image features as

$$V'_k = \sum_{j=1}^7 \sum_{i=1}^7 W_{i,j} V_{i,j,k}^* \quad (30)$$

where V^* represents the image features; (i, j) denotes the position of the unit on

the feature map of image features; k is the index for the feature map; V' represents the refined image features.

3.3.2 Fusion Network

In order to effectively fuse the image features and question features, a hierarchical fusion network is proposed. The architecture of the fusion network is shown in Fig. 15, where Q_L represents the question features extracted by LSTM network; Q_B denotes the question features extracted by bag-of-words model; V' represents the refined image features. The motivation of the hierarchical fusion network is based on the observation that different regions of the image could represent useful information in answering certain types of the questions. Therefore, a hierarchical structure is designed to fuse the features at multiple levels for effective feature integration. Let's consider a simplified example, if the given question is "how many people are in the photo," the objective of the first layer is to combine the features to identify the people in the image. Then, the second layer will further analyze the number of the identified people.

In the first layer of the fusion network, the combination of a learnable transformation and a non-linear tanh function is used to transform the features into a common feature space to facilitate the feature fusion. Specifically, for the question features, the transformations are shown in Eq. (31) and Eq. (32).

$$Q_{TL} = \tanh(W_{QL} \cdot Q_L + b_{QL}) \quad (31)$$

$$Q_{TB} = \tanh(W_{QB} \cdot Q_B + b_{QB}) \quad (32)$$

where Q_{TL} and Q_{TB} denote the transformed question features, respectively. In Eq. (31) and Eq. (32), W_{QL} , W_{QB} , b_{QL} , and b_{QB} are the weights matrix and the bias vector for the learnable transformation, respectively. These weights and bias will

Figure 15: Architecture of the hierarchical fusion network.

be optimized during the training process to learn a good transformation. Similarly, image features V' is transformed as

$$V_T = \tanh(W_V \cdot V' + b_V) \quad (33)$$

where V_T denotes the transformed image features; W_V and b_V are the weights matrix and the bias vector for the learnable transformation, respectively.

After the feature transformation, instead of using single fusion operation, multi-head fusion operation is proposed for effective feature fusion. Specifically, the transformed features are fused by element-wise multiplication and element-wise summation, respectively. The combination of these two operations makes sure that we can effectively fuse the image features and the question features.

Eq. (34) and Eq. (35) show how to use element-wise multiplication to fuse the transformed question features Q_{TL} and Q_{TB} with the transformed image features V_T .

$$F_{ML} = V_T \odot Q_{TL} \quad (34)$$

$$F_{MB} = V_T \odot Q_{TB} \quad (35)$$

where F_{ML} and F_{MB} denote the fused features, and $\odot$ represents the element-wise multiplication operation.

Similarly, Eq. (36) and Eq. (37) show how to use element-wise summation to fuse the transformed question features Q_{TL} and Q_{TB} with the transformed image features V_T .

$$F_{AL} = V_T \oplus Q_{TL} \quad (36)$$

$$F_{AB} = V_T \oplus Q_{TB} \quad (37)$$

where F_{AL} and F_{AB} denote the refined features, and $\oplus$ represents the element-wise summation operation.

Since question features are very important for the question answering, I provide a shortcut connection for the transformed question features, which is denoted as Q_{TL} . With these fused features $F_{ML}, F_{AL}, F_{MB}, F_{AB}$, and Q_{TL} , I use signed

square root and L2 normalization to normalize them. Then, these features are combined by concatenation as as

$$H_1 = Q_{TL} \parallel F_{ML} \parallel F_{AL} \parallel F_{MB} \parallel F_{AB} \quad (38)$$

where H_1 denotes the concatenated features, and $\parallel$ represents the concatenation operation. These concatenated features are treated as the question features of the second layer.

In the second layer of the fusion network, the feature fusion techniques are similar to those in the first layer. The image features and the question features are transformed into a common feature space by the combination of a learnable transformation and a non-linear tanh function. Then, the transformed question features are fused with the transformed image features by element-wise multiplication and element-wise addition, respectively. Then, these fused features and the shortcut question features are normalized by signed square root and L2 normalization. Finally, these features are combined by concatenation.

3.4 Experimental Results

In this section, several experiments are conducted to verify the effectiveness of the proposed heterogeneous data fusion model for data-driven decision-making. This section will first introduce the experimental setup, including the VQA dataset, the implementation details of the proposed model, and preprocessing of image and question. Then, the proposed model will be evaluated and compared with other related works on the VQA dataset.

3.4.1 Experimental Setup

The proposed model is evaluate on VQA dataset [104], which is a large-scale dataset for VQA task. It has 123,287 training images and 81,434 test images from

Microsoft Common Objects in Context (MS COCO) dataset [105]. The images in MS COCO dataset contain multiple objects and rich information. Therefore, they are suitable for VQA task. In VQA dataset, each image has three questions, and there are 614,163 questions in total. These questions are collected via Amazon Mechanical Turk where workers are shown an image and are requested to ask a question about this image. These questions can be grouped into three types, including “Yes/No” question (e.g., Is this baseball player holding a wooden bat?), “Number” question (e.g., How many children does this couple have?), and “Other” question (i.e., the question could be anything, such as “Where does this road lead to?” and “What color is this man’s snowboard?”). Most questions range from 4 to 10 words, and the maximum length of the question is 26. Since the questions in VQA dataset have a diverse set of possible answers, 10 answers are gathered for each question from unique workers via Amazon Mechanical Turk. Most of these answers consist of a single word, and the percentages of these answers containing one, two, or three words are 89.32%, 6.91%, and 2.74%, respectively. An answer is correct when at least three workers provide that answer.

Before the training process, image features are extracted by a pre-trained Deep Residual Net. To reduce the training time, the image features are pre-calculated and stored. During the training process, these image features are loaded into the memory.

Before extracting question features, questions are preprocessed such that they can be processed by machines. There are four major steps. First, the top 3000 answers are obtained from the training set. Then, the questions in the training set are filtered according to these top answers. Specifically, if the answer of one question is not in the top answers, this question will be removed from the training set. The top 3000 answers cover about 92% of all the questions. Then, the question

is tokenized to get the words. Finally, a vocabulary is built for these words. The size of the vocabulary is 14007, including one unknown token. During the test, if the word does not exist in the vocabulary, it will be replaced by this unknown token.

For the fusion network, the dimension of the transformed features is 1024. For the softmax layer, the top 3000 most frequent answers are selected as answer candidates. The proposed model is optimized by RMSprop gradient descent optimization algorithm with mini-batch of 500 samples. In order to alleviate the overfitting problem, dropout [106] is used.

3.4.2 Evaluation Results on VQA Dataset

The proposed AnswerNet has obtained impressive results in generating answers to different types of questions based on the input image. For instance, Fig. 16 shows a few snapshots of the generated answers for different images and questions in the VQA dataset [104]. These results demonstrate that AnswerNet can correctly answer different types of questions which require different semantic understanding.

To provide quantified evaluation of AnswerNet, this section further evaluates and compares the accuracy of AnswerNet with some related works on the test set server organized by VQA team. This server consists of *test-dev* and *test-standard*. *Test-dev* has unlimited submission times while the submission times of *test-standard* is limited to 5.

The evaluation results on VQA *test-dev* are shown in Table 10. These models are trained on train+val set without any data augmentation. The overall accuracy of AnswerNet is 64.5%, which outperforms these related works. The accuracy for the three different types of questions are 82.9%, 39.0%, and 54.5%, respectively. As we can see, AnswerNet performs really well on the “Yes/No” questions, however,

Figure 16: Snapshots of the generated answers.

it has difficulty in counting numbers.

The performance of AnswerNet is also evaluated on VQA *test-standard*, and the evaluation results are presented in Table 11. The overall accuracy of AnswerNet is 64.5%, with 82.8% in “Yes/No”, 38.6% in “Number”, and 54.4% in “Other”. As we can see, AnswerNet outperforms these related works. These results demonstrate the effectiveness of the proposed AnswerNet.

It is worth noting that, in VQA dataset, language provides a strong prior that can result in good superficial performance. For example, for the question starting with “Is there a clock?”, 98% of the answers will be “yes”. For the question starting with “what sport is?”, 41% of the answers will be tennis. These language priors can give a false impression that machines are making progress towards the goal of understanding images correctly. Actually, they are only taking advantage of language priors to achieve high accuracy.

To balance the original VQA dataset, researchers collect complementary images such that every question in the balanced dataset is associated with not just a single image, but rather a pair of similar images. These images will result in two

Table 10: Evaluation results on VQA *test-dev*.

Model	All	Yes/No	Number	Other
LSTM Q + I [104]	53.74	78.94	35.24	36.42
deeper LSTM Q + I [104]	57.75	80.50	36.77	43.08
iBOWIMG [107]	55.72	76.55	35.03	42.62
DPPnet [108]	57.22	80.71	37.24	41.69
SMN [100]	57.99	80.87	37.32	43.12
Ask Your Neurons [109]	58.4	78.4	36.4	46.3
NMN + LSTM + FT [102]	58.60	81.20	38.00	44.00
SAN [101]	58.70	79.30	36.60	46.10
ACK + LSTM [98]	59.17	81.01	38.42	45.23
D-NMN [103]	59.40	81.10	38.60	45.50
HieCoAtt [110]	61.8	79.7	38.7	51.7
RAU [111]	63.3	81.9	39.0	53.0
MCB + Att [112]	64.2	82.2	37.7	54.8
DAN [113]	64.3	83.0	39.1	53.9
AnswerNet	64.5	82.9	39.0	54.5

different answers to the question.

AnswerNet is further evaluated on this balanced dataset, VQA2 [114]. It has approximately twice the number of image-question pairs. The evaluation results on VQA2 *test-standard* are shown in Table 12. The overall accuracy of AnswerNet on VQA2 *test-standard* is 62.36% which is better than the performance of the baselines reported in [114]. These results verify the effectiveness of the proposed AnswerNet.

Table 11: Evaluation results on VQA *test-standard*.

Model	All	Yes/No	Number	Other
LSTM Q + I [104]	54.06	79.01	35.55	36.80
deeper LSTM Q + I [104]	58.16	80.56	36.53	43.73
iBOWIMG [107]	55.89	76.76	34.98	42.62
DPPnet [108]	57.36	80.28	36.92	42.24
SMN [100]	58.24	80.80	37.53	43.48
Ask Your Neurons [109]	58.4	78.2	36.3	46.3
NMN + LSTM + FT [102]	58.70	-	-	-
SAN [101]	58.90	-	-	-
D-NMN [103]	59.40	-	-	-
ACK + LSTM [98]	59.44	81.07	37.12	45.83
HieCoAtt [110]	62.1	-	-	-
RAU [111]	63.2	81.7	38.2	52.8
DAN [113]	64.2	82.8	38.1	54.0
AnswerNet	64.5	82.8	38.6	54.4

3.5 Chapter Summary

This chapter has investigated heterogeneous data fusion, which is an important part for data-driven decision-making. Two types of data, image and natural language, have been considered in this chapter. To effectively fuse these two types of data, an attention network has been designed to measure the correlation between the input data. The image features have been refined according to their correlation to the question. Then, a hierarchical fusion network has been proposed to effectively fuse the refined image features and question features at multiple lev-

Table 12: Evaluation results on VQA2 *test-standard*.

Model	All	Yes/No	Number	Other
Prior [114]	25.98	61.20	00.36	01.17
Language-only [114]	44.26	67.01	31.55	27.37
d-LSTM+n-I [104]	54.22	73.46	35.18	41.83
MCB + Att [112]	62.27	78.82	38.28	53.36
AnswerNet	62.36	79.16	40.17	52.81

els. In the fusion network, unlike previous works that use single fusion operation, multi-head fusion has been proposed for effective feature fusion. The experimental results on VQA dataset have shown that the proposed heterogeneous data fusion model outperforms the benchmark solutions.

CHAPTER 4

Data-Driven Decision-Making for Robot-Assisted Pedestrian Regulation

4.1 Chapter Overview

After automatic feature extraction and heterogeneous data fusion, we can extract insights from the data to support the decision-making process. In this chapter, a data-driven decision-making framework based on deep reinforcement learning (DRL) is designed to solve an important problem, robot-assisted pedestrian regulation. Pedestrian regulation can prevent crowd accidents in densely populated areas thus improve crowd safety. To solve this problem, mobile robot is introduced to regulate pedestrian flows for desired collective motion through the effect of passive human-robot interaction (HRI). The aim is to find the optimal robot motion planner that maneuvers the robot to the best positions that achieves maximum pedestrian outflow. In this chapter, a data-driven control scheme that directly maps from raw image observation to robot control actions is proposed.

This chapter will first introduce the background and related works for robot-assisted pedestrian regulation. Then, this problem will be formulated as a robot motion planning problem such that it can be solved with the proposed data-driven framework. Then, the details of the proposed data-driven framework based on DRL will be presented. Finally, the effectiveness of the proposed framework will be validated under different pedestrian flow scenarios. The contributions are twofold. First, the formulated robot motion planning problem provides a new way to use HRI to optimize pedestrian flows. Second, the proposed data-driven framework achieves end-to-end robot motion planning from sensor image data to robot motion decisions. Compared with traditional methods such as adaptive dynamic programming (ADP), the proposed data-driven framework avoids hand-picking features and

utilizes CNN to learn discriminative features to achieve optimal performance.

4.2 Background and Related Works

Developing pedestrian crowd regulation approaches can help to avoid crowd accidents in densely-populated areas and for emergency evacuation. Existing work on pedestrian crowd regulation includes studies from the perspective of evacuation planning [115, 116, 117] and optimal design of facilities to improve pedestrian flows [118, 119, 120]. It was recently proposed to introduce mobile robots to influence the collective motion of pedestrian crowd through HRI [121, 122, 123, 124]. Studies have been reported on robot-assisted pedestrian regulation in scenarios such as crossing pedestrian flows [121], and uni-directional pedestrian flow in an exit corridor [122, 124].

4.2.1 Human-Robot Interaction

It has been found that human motion behavior can be implicitly influenced by passive HRI [123, 121, 124, 125], namely, the robot moves in a planned motion and the humans adjust their motion to avoid collision to the robot. In this manner, human collective motion can be modified with a moving robot that dynamically interacts with humans. The effect of passive HRI has been utilized for pedestrian regulation, where desired pedestrian collective motion was achieved with optimized robot motion. The work presented in [123] studied the effect of deploying mobile robots to affect crowd dynamics, and showed that the flow efficiency was improved with appropriately-designed robot maneuver and formation patterns. The work shed light on how to utilize passive HRI for controlling and optimizing pedestrian flows. Yamamotoa et al. analyzed the characteristics of passive HRI in a crossing pedestrian flows scenario in [121], and the results were used to find optimal robot motion that can reduce congestion in crossing flows. In [124], Jiang et al. proposed

to regulate the average velocity of an uni-directional pedestrian flow with a robot moving perpendicularly to the flow direction. Due to the effect of passive HRI, the deployed robot behaves as a “virtual gate” that slows down the flow velocity around the robot, and the average flow velocity can be regulated to a desired value by adjusting the robot velocity. A robot motion control algorithm is designed based on ADP to provide adjustable robot motion online therein.

4.2.2 Reinforcement Learning for Robot Control

In the past two decades, robot planning and control with RL has attracted considerable attention from researchers [126, 127, 128, 129, 130, 131, 132]. Beom et al. [126] studied robot navigation in uncertain environments by utilizing RL and fuzzy logic. Asada et al. [127] applied a RL framework to train a soccer robot with a look-up table. In these applications, the control policy was learned without the model of the environment. However, the proposed learning algorithms deal with discretized state and action space, thus not applicable to problems with continuous or high-dimensional state and action space. In order to overcome this drawback, neural network techniques were developed to handle continuous or high-dimensional input in robotics RL. For instance, Gaskett et al. [128] exploited a three-layer perceptron to learn the wandering and visual servoing under continuous state and action space. In [129], a fuzzy neural network enabled the robot to learn basic navigation under a changing environment. However, these approaches used shallow neural networks that have a limited capability of representation learning [130], thus operate on hand-engineered features which require in-depth domain knowledge.

The bloom of DRL techniques [50, 55, 133, 134] has inspired the development of new methodologies for robot planning and control. Chen et al. [135] reported a robot motion planner for decentralized collision avoidance in human environment.

A DRL algorithm was developed to solve the collision avoidance problem. The algorithm was fed with manually-designed features, including the positions and velocities of the robot and other agents. The representation learning capability of deep learning enables robot controllers to operate directly on raw sensory inputs. Various end-to-end learning approaches have been successfully applied in mobile robot navigation and localization. Tai et al. [136] developed a mapless motion planner that only takes the sparse laser rangefinder measurements and the target position as input. The motion planner was trained through DRL without prior demonstrations and manually-designed features. In addition to laser rangefinders, image-based visual observations were used as input to end-to-end motion planners. To name a few, Zhang et al. [137] proposed a successor-feature-based DRL approach to conduct the navigation task in simple maze-like environment by using depth images from a Kinect sensor. In [138], a visual navigation problem was studied, where a robot navigated in an indoor environment using only visual observation without a map. A DNN was developed and trained to model the end-to-end robot's policy that generated robot motion action directly from a given visual input.

Although DRL has achieved encouraging progress in robotics, its applications in social navigation where a robot influences human behaviors have not been extensively reported. Recent studies [139, 140] have used RL for pedestrian simulation and show that RL is capable of learning human behaviors. This chapter will explore applying DRL to a cutting edge robotic application, namely, learning HRI for robot-assisted pedestrian regulation. This chapter follows the idea of data-driven for robot motion planning, and develops a novel DNN-based method to learn a robot motion planner only from raw sensor image data for optimal pedestrian regulation.

4.3 Problem Formulation

Inspired by the empirical study [141] of a real-world crowd stampede incident that caused significant casualties in Mina/Makkah in 2006, this chapter will focus on the pedestrian regulation in an environment where two pedestrian flows merge from different directions going through a bottleneck area. Understanding the complex merging behavior of pedestrian flows is of profound practical importance to avoid such pedestrian crowd incidents, thus studies have been conducted to find out the underlying causes and seek for solutions [141, 142, 143, 144].

This chapter will study the optimal motion planner for the robot to move to the best positions that maximize the pedestrian outflow. Specifically, the environment setup is shown in Fig. 17. Two pedestrian flows A and B merge before the corridor and move toward the exit. The amount of instantaneous inflow A and inflow B are represented by q_1 and q_2 , respectively. The entrance of the corridor becomes a bottleneck when the merged flow exceeds the capacity of the exit corridor. The aim is to prevent this congestion using a moving robot through HRI, so that the pedestrian outflow from the corridor can be maximized. In Fig. 17, the robot workspace is represented by a green dashed rectangle, and the outflow is denoted as q_t , which is defined as the number of pedestrians passing through the exit at time step t .

A finite Markov Decision Process (MDP) with discrete time step $t = \{1, 2, \dots, T\}$ is used to formulate our robot motion planning problem. MDP provides a mathematical architecture to model a sequential decision-making process. An MDP is defined as the five-tuple $(\mathbf{X}, \mathbf{U}, \mathbf{P}(\cdot, \cdot), q(\cdot, \cdot), \gamma)$, where $\mathbf{X}$ represents the observation space of the system, $\mathbf{U}$ denotes a set of permissible actions, $\mathbf{P}(\cdot, \cdot)$ represents the observation transition model, $q(\cdot, \cdot)$ is the immediate reward, and γ denotes a discount factor. The robot motion planning problem is defined using

Figure 17: The merging pedestrian flows scenario.

these five elements as follows.

Observation space

The observation $\mathbf{x}_t$ is the image of the pedestrians in the environment at time step t . It not only contains the positions of the pedestrians, but also shows other features about the pedestrians that are essential for the robot motion planner, such as density.

Action space

In the formulation, the robot workspace is discretized as a regular grid. The robot moves on the grid points. The action space $\mathbf{U}$ contains four permissible directions of the robot motion decision $\mathbf{u}_t$, i.e., “up”, “down”, “right”, and “left”, in the grid-based map defined in the robot’s workspace.

Observation transition

The transition from observation $\mathbf{x}_t$ to the next observation $\mathbf{x}_{t+1}$ is defined as

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t). \quad (39)$$

The robot position is updated according to the robot motion decision $\mathbf{u}_t$. Then, the robot influences the pedestrian motion through HRI, and the system observation transits from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. It is challenging to model the accurate observation transition function for pedestrians because the HRI and the human motion are exceedingly complex. In this chapter, a data-driven approach is proposed to provide control action.

Reward

The reward is the instantaneous outflow q_t which is the number of pedestrians passing through the exit at time step t .

In this formulation, an action-value function, $Q_\pi(\mathbf{x}, \mathbf{u})$, is defined to assess the performance of taking a motion decision $\mathbf{u}$ when given an observation $\mathbf{x}$ under a robot motion planning policy π . This policy maps from each system observation to the probability of taking the robot motion decision. $Q_\pi(\mathbf{x}, \mathbf{u})$ is defined below as the expected sum of rewards starting from observation $\mathbf{x}$, taking the motion decision $\mathbf{u}$ by following policy π , that is,

$$Q_\pi(\mathbf{x}, \mathbf{u}) = \mathbb{E}_\pi \left[\sum_{k=0}^K \gamma^k q_{t+k} \middle| \mathbf{x}_t = \mathbf{x}, \mathbf{u}_t = \mathbf{u} \right], \quad (40)$$

where K represents the number of future time steps, $\mathbb{E}_\pi[\cdot]$ denotes the expected value given that the robot follows policy π , and γ balances the importance between the future rewards and the immediate reward. In this chapter, γ is set to 1, which means the future rewards have the same importance as the immediate reward. Since q_t is the instantaneous outflow, $Q_\pi(\mathbf{x}, \mathbf{u})$ is equivalent to the accumulated outflow in the horizon of K time steps.

The objective of the robot motion planning problem is to determine the optimal

policy π^* which maximizes the expected sum of rewards as

$$Q^*(\mathbf{x}, \mathbf{u}) = \max_{\pi} Q_{\pi}(\mathbf{x}, \mathbf{u}), \quad (41)$$

where $Q^*(\mathbf{x}, \mathbf{u})$ represents the optimal action-value function.

Thus, following the optimal policy π^* and starting from an arbitrary initial observation $\mathbf{x}_1$, we can maximize the number of pedestrians passing through the exit in the horizon of K time steps, i.e., $Q^*(\mathbf{x}_1, \mathbf{u}) = \max \sum_{k=0}^K q_{k+1}$.

4.4 Data-Driven Robot Motion Control

Traditional approaches rely on feature extraction that is designed with experience. This feature engineering process may neglect some important features since finding all useful features requires thorough insight into the problem. These features, however, can be easily observed in the images of the environment, which have rich information of state of the robot and the pedestrians. Also, for traditional methods, it is challenging to find optimal robot motion policy π^* using high dimensional images. To tackle these challenges, a data-driven framework based on DRL is proposed [61].

4.4.1 Overall Architecture

The overall framework of the proposed data-driven robot motion control for pedestrian regulation is illustrated in Fig. 18. A simulated environment is used to validate the effectiveness of the proposed approach. The image of the simulated environment is passed into a DNN to approximate the action-value of robot motion decision $\mathbf{u}$. The decision $\mathbf{u}_i$ with the highest action-value is selected for the robot. It is worth noting that the proposed data-driven approach can directly learn the optimal robot motion control policy from the image.

The architecture of the DNN shown in Fig. 18 consists of a CNN and a Q network. The details of these two networks are presented in this section.

Figure 18: The overall framework of the proposed data-driven robot motion control for pedestrian regulation.

A two-layer CNN is implemented to extract discriminative features from the image to facilitate the robot motion planning process. The convolutional layers (Conv 1 and Conv 2 in Fig. 18) have several feature maps which are represented by green and blue squares in the figure. The feature map in Conv 1 layer is obtained by applying a convolution operation to the input image. Similarly, the feature map in Conv2 layer is obtained by applying convolution operation to the Conv1 layer. Then, these feature maps are flattened into a vector, $\mathbf{d}$.

The features extracted by the CNN are fed into the Q network, a three-layer fully-connected neural network. This type of neural network with a finite number of hidden units can uniformly approximate continuous functions [145]. The value of hidden unit can be calculated as

$$\mathbf{h} = g(\mathbf{W}_h * \mathbf{d} + \mathbf{b}_h), \quad (42)$$

where $\mathbf{W}_h$ and $\mathbf{b}_h$ represent the weights and the bias, respectively. Then, the hidden layer is connected with the output layer. The value of the output unit is the estimation of the action-value for robot motion decision $\mathbf{u}$ when given the

input observation $\mathbf{x}$, i.e.,

$$Q(\mathbf{x}, \mathbf{u}) = g(\mathbf{W}_o * \mathbf{h} + \mathbf{b}_o) \quad (43)$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ represent the weights and the bias, respectively. Finally, the robot motion decision with the largest action-value is outputted, i.e., $\mathbf{u} = \operatorname{argmax}_{\mathbf{u} \in U} Q(\mathbf{x}, \mathbf{u})$.

4.4.2 Training Process

Algorithm 1 illustrates how to train the proposed model according to double Q-learning [146]. The input of Algorithm 1 is the image observation $\mathbf{x}$ and the reward q , respectively. Its output is the DNN's parameters θ . Parameters θ include the weights and bias of the CNN and the Q network.

In Algorithm 1, the DNN in Fig. 18 is referred to as the main DNN. In line 1, the main DNN's parameters θ are randomly initialized. Then, a target DNN is initialized, and its parameters $\bar{\theta}$ are cloned from the parameters of the main DNN. In the outer loop starting from line 3, the parameters θ are iteratively updated through N epoches. At the beginning of each epoch, the environment is initialized, i.e., the pedestrian and the robot positions are initialized. Then, the initial image observation $\mathbf{x}_1$ is obtained. In the inner loop starting from line 5, the environment evolves for T time steps. At each time step, from line 6 to line 11, an ϵ -greedy search method [52] is applied to choose the action $\mathbf{u}_t$. Specifically, a random number c is generated from a uniform distribution on the interval $(0, 1)$. If c is larger than a constant ϵ , the action $\mathbf{u}_t$ is generated by the main DNN in Fig. 18. Otherwise, the action $\mathbf{u}_t$ is randomly chosen from the action set U . After that, in line 12, the robot position is updated according to the action $\mathbf{u}_t$; the reward q_t (instantaneous outflow) is observed; and a new observation $\mathbf{x}_{t+1}$ is obtained. Then, in line 13, the experience tuple $(\mathbf{x}_t, \mathbf{u}_t, q_t, \mathbf{x}_{t+1})$ is stored in a buffer $\mathcal{B}$. From

Algorithm 1 Training of Deep Neural Network

Input: Image observation $\mathbf{x}$ and reward q

Output: DNN's parameters θ

```
1: Randomly initialize main DNN's parameters  $\theta$ .
2: Initialize a target DNN with parameters  $\bar{\theta} = \theta$ .
3: for Epoch=1:N do
4:   Initialize the environment and obtain observation  $\mathbf{x}_1$ .
5:   for Time step  $t = 1:T$  do
6:     Sample  $c$  from a uniform distribution  $\mathcal{U}(0, 1)$ .
7:     if  $c > \epsilon$  then
8:       Obtain  $\mathbf{u}_t$  from the main DNN.
9:     else
10:      Randomly select  $\mathbf{u}_t$  from the action set  $\mathbf{U}$ .
11:    end if
12:    Update robot position  $\mathbf{p}_t$ , observe reward  $q_t$ , and obtain the next image
    observation  $\mathbf{x}_{t+1}$ .
13:    Store the tuple  $(\mathbf{x}_t, \mathbf{u}_t, q_t, \mathbf{x}_{t+1})$  in buffer  $\mathcal{B}$ .
14:    Randomly sample a batch of tuples  $\{(\mathbf{x}_j, \mathbf{u}_j, q_j, \mathbf{x}_{j+1})\}_{j=1}^D$  from  $\mathcal{B}$ .
15:     $y_j \leftarrow q_j + \gamma Q(\mathbf{x}_{j+1}, \arg\max_{\mathbf{u}} Q(\mathbf{x}_{j+1}, \mathbf{u}; \theta_t); \bar{\theta})$ .
16:    Calculate the loss function  $L(\theta_t) = \frac{1}{D} \sum_{j=1}^D [y_j - Q(\mathbf{x}_j, \mathbf{u}_j; \theta_t)]^2$ .
17:    Update parameters  $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L(\theta_t)$ 
18:    Every  $S$  steps reset  $\bar{\theta} = \theta$ .
19:  end for
20: end for
```

line 14 to line 17, the main DNN's parameters θ are optimized. Specifically, in line 14, a batch of tuples $\{(\mathbf{x}_j, \mathbf{u}_j, q_j, \mathbf{x}_{j+1})\}_{j=1}^D$ are randomly sampled from the buffer $\mathcal{B}$. With these tuples, in line 15, the target action-value y_j is calculated as

$$y_j = q_j + \gamma Q\left(\mathbf{x}_{j+1}, \arg\max_{\mathbf{u}} Q(\mathbf{x}_{j+1}, \mathbf{u}; \theta_t); \bar{\theta}\right). \quad (44)$$

Then, in line 16, the loss function can be derived as

$$L(\theta_t) = \frac{1}{D} \sum_{j=1}^D [y_j - Q(\mathbf{x}_j, \mathbf{u}_j; \theta_t)]^2, \quad (45)$$

which is the mean squared error between the target action-value y_j and the action-value $Q(\mathbf{x}_j, \mathbf{u}_j; \theta_t)$ estimated by the main DNN. Then, the loss function is mini-

mized by updating the main DNN's parameters θ according to gradient descent rule, i.e., $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L(\theta_t)$, where η denotes the learning rate and $\nabla_{\theta_t} L(\theta_t)$ is the gradient of the loss function. Then, at every S steps, the target DNN's parameters are reset as $\bar{\theta} = \theta$. After the training process, the main DNN's parameters θ will be outputted for the online robot motion planner deployment.

During the training process, Adam optimizer is used to minimize the loss function by updating the parameters of DNN. The learning rate is set as 0.0001. The exponential decay rates for the first and the second moment estimates are set as 0.9 and 0.999, respectively. The rectified linear unit (ReLU) activation function is used in the DNN. The size of the mini-batch is 32.

4.4.3 Online Deployment

After the training process in Algorithm 1, the parameters of the DNN will be fixed for the online robot motion planner deployment.

The deployment algorithm is summarized in Algorithm 2. Its input is the images of the simulated environment, and its output is the robot motion decisions. In line 1 of Algorithm 2, I first load the DNN's parameters θ trained by Algorithm 1. Then, the robot position $\mathbf{p}_0$ can be initialized as any position in the robot workspace. In the loop starting from line 3, the robot is controlled by the DNN to regulate the pedestrians for T time steps. The image of the environment is obtained from the simulated environment. Then, this image is fed into the CNN to extract discriminative image features. After that, in line 6, these features are fed into the Q network to calculate action-value $Q(\mathbf{x}_t, \mathbf{u}; \theta)$ for all permissible robot motion decisions. Then, in line 7, the decision $\mathbf{u}_t$ is selected as $\mathbf{u}_t = \operatorname{argmax}_{\mathbf{u} \in U} Q(\mathbf{x}_t, \mathbf{u}; \theta)$. Finally, $\mathbf{u}_t$ is outputted to the robot.

Algorithm 2 Robot Motion Planning

Input: Images of the environment

Output: Robot motion decisions $\mathbf{u}_{1:T}$

- 1: Load the DNN’s parameters θ trained by Algorithm 1.
 - 2: Initialize robot position $\mathbf{p}_0$.
 - 3: **for** Time step $t = 1:T$ **do**
 - 4: Obtain the image of the environment $\mathbf{x}_t$.
 - 5: CNN extracts features from the image.
 - 6: Q network calculates action-value $Q(\mathbf{x}_t, \mathbf{u}; \theta)$.
 - 7: $\mathbf{u}_t \leftarrow \operatorname{argmax}_{\mathbf{u} \in U} Q(\mathbf{x}_t, \mathbf{u}; \theta)$
 - 8: Output robot motion decision $\mathbf{u}_t$.
 - 9: **end for**
-

4.5 Experimental Results

The effectiveness of the proposed data-driven robot motion control is verified through the simulation experiments. This section will first introduce the simulation setup, and then present the HRI characteristic results which serve as the ground-truth for algorithm validation. After that, the performance of the proposed approach will be evaluated with different robot initial positions and pedestrian inflow conditions. The performance will also be evaluated under the scenario where the pedestrian inflow changes. Finally, this section will visualize the hidden features extracted by the Q-network and demonstrate how these features contribute to the success of the proposed data-driven approach.

4.5.1 Experimental Setup

The Environment

A simulation environment is developed for the merging pedestrian flow scenario. The image used to train the DNN is generated from the simulated environment and is shown in Fig. 19. The dimension of this image is 200×200 . The pedestrians in flow A and flow B are represented by red and blue circles in this simulation environment. The robot is denoted by a black square. The size of the environment is 8 m by 8 m with an exit corridor of $w = 4$ m in width, that is

Figure 19: Snapshot of the environment used in the simulation.

$x \sim [4, 8]$ m and $y \sim [4, 8]$ m. The environment space is continuous for the pedestrians. The robot moves on the grid points of a regular grid defined in $x \sim [0, 4]$ m and $y \sim [0, 8]$ m. The grid size is 0.2 m. The instantaneous outflow q_t is averaged over 5 most recent measurements taken at every second to reduce the noise of the observation.

Pedestrian Motion Simulation

The existing Social Force Model (SFM) [141] with HRI forces is used to simulate the pedestrian's motion and HRI. SFM is a commonly used model to simulate pedestrian's behavior, and has been used to simulate pedestrian's motion in recent studies on robot navigation [147, 148, 124]. In SFM, the pedestrian is represented by plane circle with mass m_i . The motion dynamics of pedestrians are governed by the self-driven force and the interaction forces exerted on a pedestrian by the environment including other pedestrians, the boundary wall, and the robot. The robot is a passive element in the SFM, i.e., the robot does not suffer a repulsive force from the pedestrians. Using robot as a passive element has been validated

by real human-robot experiments [149]. The details of SFM and its parameters are shown in [61]. The pedestrian’s initial speed is set as 2 m/s, and the pedestrian’s position is randomly initialized using a uniform distribution $\mathcal{U}(a, b)$. For pedestrian in flow A, $x \sim \mathcal{U}(-5, -3)$ m and $y \sim \mathcal{U}(4.5, 7.5)$ m. For pedestrian in flow B, $x \sim \mathcal{U}(0.5, 3.5)$ m and $y \sim \mathcal{U}(-5, -3)$ m. The radius of the robot is set to 0.2 m.

Simulation Scenarios

Extensive simulations with random robot initial positions and different pedestrian flows have been conducted. The inflow ratios between flow A and flow B is denoted as q_1/q_2 . The sum of the number of pedestrians in flow A and flow B is set to be 300, and the inflow ratio q_1/q_2 is changed to create different cases. This section will first present two simulation cases with different flow ratios q_1/q_2 . In Case 1, $q_1/q_2 = 1/3$. In Case 2, $q_1/q_2 = 2/1$. In each case, different robot initial positions will be chosen to validate whether the robot can find a good motion planning policy such that the accumulated outflow, $\sum_{t=0}^T q(t)$, is maximized. The duration of each simulation run is set as $T = 400$ s. Then, this section will perform the simulation of Case 3 where the initial inflow ratio of flow A and flow B is $q_1/q_2 = 5 : 1$, and then it changes to $q_1/q_2 = 1 : 11$ at $t = 300$ s. The duration of the simulation run is set to 600 s. The robot is expected to adjust its position accordingly to maximize the accumulated outflow under these two flow ratios.

DNN Structure

The filter size of the Conv1 layer in CNN is 8×8 . The output of the Conv1 layer is 16 feature maps, each of which is of dimension 49×49 . The filter size of the Conv2 layer in CNN is 4×4 . The output of the second layer is 32 feature maps, each of which is of dimension 23×23 . These feature maps are flattened

into a 16928-dimensional vector and fed into the Q network. The hidden layer and output layer of the Q network have 256 and 4 units, respectively.

Computer Configuration and Implementation

The training and testing of the proposed approach are conducted on a workstation with one 12-core i7-6800K CPU and two NVIDIA TITAN Xp GPUs. The DNN is implemented with Python on GPU 0 while the pedestrian motion model is implemented with PyCUDA on GPU 1.

4.5.2 HRI Characteristics

Before verifying the proposed data-driven approach, the effect of the HRI on the pedestrian accumulated outflow is analyzed by performing the simulations with the robot placed at different fixed positions. Specifically, a set of 800 robot positions is chosen from the region defined in $x \sim [0, 3.8]$ m and $y \sim [0, 7.8]$ m on the grid map with the grid size of 0.2 m. In each run, the robot is placed at one of the positions and the accumulated outflow at $T = 400$ s is recorded. For each position in the set, 5 runs are repeated and the average accumulated outflow over these 5 runs is calculated. Fig. 20a and Fig. 20b illustrate the top view and 3D view of the HRI characteristics results for Case 1, respectively. The quantity of the accumulated outflow of each position is indicated according to the color bar on the right, ranging from dark red (high accumulated outflow) to dark blue (low accumulated outflow). To better show the difference of the positions in accumulated outflow, the range of accumulated outflow value is adjusted to be mapped on the color map. Thus, accumulated outflow values that are lower than the minimum of the range are indicated by the same color (i.e., dark blue). The robot positions with the highest accumulated outflow are marked by a rectangle in Fig. 20a.

Figure 20: HRI characteristics for Case 1.

Similarly, the HRI characteristics results for Case 2 are presented in Fig. 21. The robot positions with the highest accumulated outflow are marked by an ellipse in Fig. 21a.

Figure 21: HRI characteristics for Case 2.

The HRI characteristics results for the two inflow ratios in Case 3 are illustrated in Fig. 22. The ellipse in Fig. 22a and the rectangular in Fig. 22b highlight the robot positions with highest accumulated outflow.

Fig. 20 demonstrates that under the inflow ratio $q_1/q_2=1:3$ (which represents that the main flow A is less than the branch flow B), the best position for the

Figure 22: HRI characteristics for Case 3.

robot to stay is the region located on the right of the branch channel (flow B). Fig. 21 shows that under the inflow ratio $q_1/q_2=2:1$, the best position for the robot is the region where the two flows merge. Similar region of the best robot positions can be found in Fig. 22a for inflow ratio $q_1/q_2=5:1$. Comparing Fig. 20a with Fig. 21a, we can observe that the optimal regions for Case 1 and Case 2 are different. It is intuitive that when the main flow A is bigger in Case 2, congestion may occur in the merging area, and the robot should stay in the position to impede more pedestrians from getting into the bottleneck to prevent congestion. On the contrary, when the branch flow B is bigger in Case 1, the robot should stay away from the middle of the flow to have more people passing through to maximize the overall pedestrian outflow. It is worth noting that when the branch flow B keep getting bigger, the inflow condition can reach to the extent that nearly all the incoming pedestrians are in flow B, which is represented by the inflow ratio $q_1/q_2=1:11$ in Case 3. Fig. 22b shows that in this case the robot should stay away from flow B, thus to keep flow B as smooth as possible. The HRI characteristics shows that optimal robot positions exist to maximize the outflow of pedestrians, and thus are used as “ground-truth” to validate whether the proposed approach

can learn the optimal robot motion planning such that the accumulated outflow is maximized under different inflow ratios.

4.5.3 Performance Evaluation

Evaluation Results for Merging Pedestrian Flows

The training performance is presented by running Algorithm 1 in this section. The DNN is trained for 3500 epochs to learn the optimal robot motion policy. In each epoch, the inflow ratio q_1/q_2 is randomly chosen from the set $(5 : 1, 2 : 1, 1 : 1, 1 : 2, 1 : 3, 1 : 11)$, which represent different pedestrian distributions in the two merging flows. The robot initial position is randomly chosen from the grid point. The simulation time for each epoch is 400 s. The evolution of the accumulated outflow at $T = 400$ s over 3500 epochs is shown in Fig. 23. In the first 250 epochs, the robot motion decision is randomly chosen from the 4 permissible directions. Then, from epoch 250 to epoch 500, the robot motion decision is randomly selected with a probability ϵ , otherwise, it is generated by the proposed DNN. In this phase, the probability ϵ gradually reduces from 1.0 to 0.1, and keeps 0.1 afterward. In Fig. 23, we can observe that the accumulated outflow increases steadily before 2000 epochs, and then it converges with small oscillations. This training result shows that the proposed approach succeeds in learning a robot motion policy to maximize the accumulated outflow under different robot initial positions and different inflow ratios.

After the training process, the robot motion planning results are obtained by running Algorithm 2. The proposed approach is evaluated under Case 1 and Case 2. In each case, extensive simulations with random initial robot positions were conducted.

In case 1, the proposed data-driven robot motion control is evaluated under the pedestrian flow $q_1/q_2=1:3$. The results of Case 1 with robot initial position

Figure 23: The evolution of the accumulated outflow over 3500 epochs during the training process.

[0.4, 2] are illustrated in Fig. 24. The robot trajectory is shown in Fig. 24a, where the robot moves in the grid-based map defined in the robot's workspace. The robot initial position is denoted by a red star, and the robot positions at different time steps are represented by black stars. The arrow at the star indicates the robot motion direction generated by the proposed approach. The time history of robot position in x and y directions are presented in Fig. 24b. We can observe that after around 20 s, the robot converges into a region, $x \sim [2.6, 3.6]$ and $y \sim [2.6, 3.8]$, which is the optimal region as verified in HRI Characteristics and marked by the rectangular in Fig. 20a. Fig. 24c shows the instantaneous outflow, q_t , where the black curve shows the result with the proposed regulation strategy, and the red curve shows the result without robot regulation. One can see that the instantaneous outflow, q_t , with the proposed regulation is higher than that without robot regulation. Furthermore, in the green box, there is a significant drop in the red curve, which is caused by congestion without the robot regulation. It can be seen that the congestion is avoided by the proposed robot regulation, and the black curve is relatively more smooth without sharp drops. The accumulated outflow,

Figure 24: Robot initial position $[0.4, 2]$ for Case 1.

$\sum_{t=0}^T q_t$, at $T = 400$ s with and without robot are 6522 and 5950, respectively. With robot regulation, the accumulated outflow increases by 9.61%.

Similarly, Fig. 25 illustrates the results of Case 1 with robot initial position $[3.4, 7.4]$. We can see that the robot learns to avoid the region in Fig. 20a that results in low accumulated outflow. After around 100 s, the robot converges into the optimal region. The accumulated outflow at $T = 400$ s with and without robot

Figure 25: Robot initial position $[3.4, 7.4]$ for Case 1.

are 6313 and 5950, respectively. With robot regulation, the accumulated outflow increases by 6.10%.

Statistical results are also provided to demonstrate the effectiveness of the proposed approach. The performance of the proposed approach is compared with no robot, a randomly-moving robot, and ADP-control robot. For the randomly-moving robot, the robot motion is randomly selected from the permissible actions.

For the ADP-control robot, the robot motion is generated by ADP which has been successfully used in [124] for pedestrian regulation. The simulation is repeated 10 times for each initial position. The average accumulated outflow at $T = 400$ s is calculated over 10 runs. The simulation results for Case 1 under robot initial positions $[0.4, 2]$ and $[3.4, 7.4]$ are shown in Fig. 26. Without robot, the average accumulated outflow is 5989, and its standard deviation is 73. With randomly moving robot, the average accumulated outflow under these two initial positions are 6198 and 4650, respectively. Their standard deviations are 240 and 514, respectively. With ADP-control robot, the average accumulated outflow under these two initial positions are 6286 and 6143, respectively. Their standard deviations are 35 and 37, respectively. With the proposed DRL control, the average accumulated outflow increase to 6475 and 6310, respectively. Their standard deviations are 53 and 36, respectively. We can see that the proposed approach greatly improves the accumulated outflow compared to the no-robot case, the randomly-moving robot case, and the ADP-control robot case.

In order to extensively evaluate the performance of the proposed approach, the results of 10 different robot initial positions are shown in Fig. 27 for Case 1. These initial positions are randomly selected from the boundary of the robot workspace, which are away from the optimal region marked in Fig. 20a. The shadow area highlights the position range from maximum to minimum of the 10 trajectories at each time step, and the blue solid line illustrates the average position of these trajectories at each time step. We can observe that the robot can converge to the optimal region from different initial positions.

In case 2, the proposed data-driven robot motion control is evaluated under the pedestrian flow $q_1/q_2 = 2/1$. The results of Case 2 with robot initial position $[0.4, 2]$ are shown in Fig. 28. Fig. 28a and Fig. 28b show that the robot converges

Figure 26: Average accumulated outflow for Case 1 under robot initial positions $[0.4, 2]$ and $[3.4, 7.4]$ over 10 runs.

Figure 27: Time history of robot position under 10 robot initial positions for Case 1.

into the optimal region marked by ellipse in Fig. 21a after around 20 s. One can see from Fig. 28c that the instantaneous outflow with robot regulation is higher than that without robot regulation. In the green box, there are sharp drops in the red curve that indicates severe crowd congestion. The accumulated outflow with and without robot are 6509 and 5989, respectively. With robot regulation, the

Figure 28: Robot initial position $[0.4, 2]$ for Case 2.

accumulated outflow increases by 8.68%.

Similarly, the results of Case 2 with robot initial position $[3.4, 7.4]$ are shown in Fig. 29. From Fig. 29a, one can observe that the robot learns to avoid the region in Fig. 21a that results in low outflow. In addition, the robot converges into the optimal region after around 50 s. Fig. 29c shows the improvement of instantaneous flow. The accumulated outflow with and without robot are 6356

(a) Robot trajectory

(b) Robot position in x and y directions

(c) Instantaneous outflow

Figure 29: Robot initial position $[3.4, 7.4]$ for Case 2.

and 5989, respectively. With robot regulation, the accumulated outflow increases by 6.13%.

The statistical results for Case 2 under robot initial positions $[0.4, 2]$ and $[3.4, 7.4]$ are shown in Fig. 30. The simulation is repeated 10 times for each initial position. The average accumulated outflow at $T = 400$ s is calculated over 10 runs.

Figure 30: Average accumulated outflow for Case 2 under robot initial positions $[0.4, 2]$ and $[3.4, 7.4]$ over 10 runs.

Without robot, the average accumulated outflow is 6027, and its standard deviation is 33. With randomly moving robot, the average accumulated outflow under these two initial positions are 6060 and 4796, respectively. Their standard deviations are 69 and 821, respectively. With ADP-control robot, the average accumulated outflow under these two initial positions are 6223 and 6129, respectively. Their standard deviations are 27 and 18, respectively. With the proposed DRL control, the average accumulated outflow increase to 6485 and 6347, respectively. Their standard deviations are 17 and 30, respectively. We can see that the proposed approach greatly improves the accumulated outflow compared to these benchmark solutions.

In addition, the results of 10 robot initial positions are presented in Fig. 31 for Case 2. These initial positions are randomly selected from the boundary of the robot workspace, which are away from the optimal region marked in Fig. 21a. The shadow area highlights the position range from maximum to minimum of the

Figure 31: Time history of robot position under 10 robot initial positions for Case 2.

10 trajectories at each time step, and the solid blue line illustrates the average position of these trajectories at each time step. We can observe that the robot can converge to the optimal region marked in Fig. 21a from different initial positions.

These results demonstrate that the proposed data-driven approach succeeds in learning the optimal robot motion planning such that the accumulated outflow is maximized under different inflow ratios and different robot initial positions.

The computational time and resource usage for the training phase and the testing phase are presented in Table 13. It takes about 30 hours to train the DNN on the workstation. The GPU memory usage are 4.44% and 2.64% for GPU 0 and GPU 1, respectively. The CPU memory usage is 2.07%. For the online robot motion planner running Algorithm 2, the image of the environment is fed into the robot every 1 second. It takes about 125 ms for Algorithm 2 to generate the robot motion decision after receiving the input image. The GPU memory usage are 4.44% and 2.64%, respectively. The CPU memory usage is 1.49%. We can see that after the training, the robot motion planner is fast enough for online control.

Table 13: Computational time and resource usage

	Time	GPU 0 memory	GPU 1 memory	CPU memory
Training	30 hours	4.44%	2.64%	2.07%
Online	125 ms	4.44%	2.64%	1.49%

Evaluation Results for Changing Pedestrian Flows

In this section, the performance of the proposed data-driven approach is evaluated when the pedestrian inflow ratio changes. The simulation is conducted under Case 3 where the initial inflow ratio of flow A and flow B is $q_1/q_2 = 5 : 1$, and then it changes to $q_1/q_2 = 1 : 11$ at $t = 300$ s. Case 3 represents the scenario where the main inflow (flow A) is more than the branch inflow (flow B) initially and then the main inflow gets less than the branch inflow. The robot is expected to adjust its position accordingly to maximize the accumulated outflow under these two flow ratios. Fig. 22a and Fig. 22b show the HRI characteristics of inflow ratios $q_1/q_2 = 5 : 1$ and $q_1/q_2 = 1 : 11$, respectively. The ellipse in Fig. 22a and the rectangular in Fig. 22b highlight the robot positions with highest accumulated outflow. The robot is expected to converge into the ellipse region in Fig. 22a for $q_1/q_2 = 5 : 1$ and then adjust to the rectangular region in Fig. 22b for $q_1/q_2 = 1 : 11$.

Fig. 32 shows the simulation results for Case 3 where the robot initial position is $[1, 1]$. We can observe that the robot converges to the ellipse region in Fig. 22a at about 30s. When the inflow ratio changes at 300s, the robot adjust its position and converges to the rectangular region in Fig. 22b after 340s. These results verify the proposed approach can replan the robot motion in real time according to the change of the pedestrian inflows.

Figure 32: Evaluation results for Case 3.

Evaluation results in 3D environment

In this section, the proposed approach is evaluated in a 3D continuous environment constructed by Unity 3D engine. The new 3D environment is shown in Fig. 33a. The pedestrians in flow A and flow B are represented by red and blue, respectively. The 3D pedestrian model is shown in Fig. 33b.

The evaluation results under pedestrian flow $q_1/q_2=1:3$ are shown in Fig. 34. The robot trajectory is shown in Fig. 34a. The robot initial position is denoted by a red star, and the robot positions at different time steps are represented by black stars. The arrow at the star indicates the robot motion direction generated by the proposed approach. The time history of robot position in x and y directions are presented in Fig. 34b. We can observe that after around 20 s, the robot converges into a region, $x \sim [2.8, 3.8]$ and $y \sim [2.0, 3.6]$, which is in the optimal region as verified in HRI Characteristics and marked by the rectangular in Fig. 20a. Fig. 34c shows the instantaneous outflow, q_t , where the black curve shows the result with the proposed regulation strategy, and the red curve shows the result without

(a) Snapshot of the 3D environment

(b) 3D pedestrian model

Figure 33: The 3D environment constructed by Unity 3D engine.

robot regulation. One can see that the instantaneous outflow, q_t , with the proposed regulation is higher than that without robot regulation. Furthermore, in the green box, there is a significant drop in the red curve, which is caused by congestion without the robot regulation. It can be seen that the congestion is avoided by the proposed robot regulation, and the black curve is relatively more smooth without sharp drops. The accumulated outflow, $\sum_{t=0}^T q_t$, at $T = 400$ s with and without robot are 6549 and 5950, respectively. With the proposed robot regulation, the accumulated outflow increases by 10.07%. These results verify the effectiveness of our proposed approach in this 3D environment.

4.5.4 Analysis of the Extracted Hidden Features

This section analyzes the hidden representations of the Q network (i.e., the vector $\mathbf{h}$ in Eq. (42)) and demonstrates how these representations contribute to the successful performance on this pedestrian-flow regulation problem. The experiment is conducted under Case 1 where inflow ratio $q_1/q_2 = 1 : 3$. By feeding an image of the environment, $\mathbf{x}$, into the DNN, we can get a 256-dimensional vector

(a) Robot trajectory

(b) Robot position in x and y directions

(c) Instantaneous outflow

Figure 34: Evaluation results in the 3D environment.

from the hidden layer of the Q network and the maximum output of the Q network, which is denoted as $V = \max_u Q(\mathbf{x}, \mathbf{u}; \theta)$. 400 samples of the 256-dimensional vector and their V values under Case 1 are collected. In Fig. 35, the high-dimensional vector is projected onto a two-dimensional space by a technique called t-distributed stochastic neighbor embedding (t-SNE) [150]. The 256-dimensional vectors with small Euclidean distance in the high-dimensional space will be projected to nearby points in the two-dimensional space. The points are colored with triangles accord-

Figure 35: Two-dimensional t-SNE embedding of the hidden representations of the Q network.

ing to their V values ranging from dark red (highest V) to drake blue (lowest V). In Fig. 35, some input images corresponding to a selected number of points are presented. We can observe that the four input images on the bottom correspond to high V values while the four input images on the top correspond to low V values. By comparing the robot position (black square) in these images with the HRI characteristics, we can find that the robot positions in the bottom input images achieve high accumulated outflow while the robot positions in the top input images achieve low accumulated outflow. These results demonstrate the DNN has learned the relationship between the robot position and the accumulated outflow. Furthermore, we can see that the points with similar colors are mapped nearby while the points with different colors are mapped far-away. These results show

that the DNN has learned to extract suitable features from the input image to support the decision-making process.

4.6 Chapter Summary

This chapter has proposed a data-driven decision-making framework for robot-assisted pedestrian regulation. A mobile robot has been introduced to regulate pedestrian flows through passive HRI. The proposed data-driven robot motion control can directly generate the robot control actions with raw image observation. The effectiveness of the proposed data-driven robot motion control has been verified through simulation experiments under different pedestrian flow conditions. The experimental results have shown that the proposed approach has learned optimal robot motion planning to maximize the accumulated outflow under different inflow ratios and different robot initial positions. In addition, the visualization of the extracted hidden features demonstrates the effectiveness of the automatic feature extraction.

This chapter mainly focused on the formulation of the robot-assisted pedestrian regulation problem and the novel data-driven approach to solve it. One potential future work is to implement the proposed approach in real-world scenarios. The future work may include the following parts: set up an environment with an installed surveillance camera and a pedestrian tracking system; collect real images of pedestrian flows from the surveillance camera; transmit images and instantaneous outflow data to a mobile robot via a local area network; use the proposed approach to calculate motion decisions.

CHAPTER 5

Data-Driven Decision-Making for EV Charging Management

5.1 Chapter Overview

Data-driven decision-making has achieved promising results in the previous chapter for the pedestrian regulation problem. This chapter will focus on developing data-driven approach for the EV charging management problem in smart grid. Efficiently managing the EV charging schedules can not only reduce customer's charging cost but also contribute to peak-shaving and valley-filling for the power grid. A data-driven approach is proposed to find the optimal charging policy that can shift the charging schedules from peak-hours to off-peak hours and make sure the EV can be fully charged upon departure.

In this chapter, first, the background and related works for this EV charging management problem will be discussed. Then, the EV charging management problem will be formulated as an MDP. Then, the architecture of the proposed data-driven approach based on DRL will be introduced. Finally, the effectiveness of the proposed approach will be verified in numerous simulations. The contributions are twofold. First, an MDP with unknown transition probability is constructed from the user's perspective to formulate the EV charging management problem. The randomness of the electricity price and the commuting behavior is taken into consideration to formulate the real-world scenarios. Second, a DRL based data-driven approach, which does not require any system model information, is proposed to determine an optimal strategy for this real-time scheduling problem.

5.2 Background and Related Works

EV provides a sustainable alternative to fossil-fuel vehicles and can significantly reduce the transport-related pollution. Another benefit of EVs is the cost

reduction for consumers since it is cheaper to charge an EV than fill up with gasoline [151]. As the real-time electricity price has been adopted by many utility companies to encourage shifting energy usage to off-peak hours [152], the charging cost can be reduced by optimizing the charging schedules [153]. In addition, an EV can discharge energy back to the electric grid by working in the vehicle-to-grid (V2G) mode and thereby make money [154].

Due to the existence of randomness in traffic conditions, user's commuting behavior, and pricing process of the utility, EV arrival and departure time, EV energy consumption, and electricity prices are dynamic and time-varying. Therefore, efficiently managing EV charging/discharging to reduce the cost becomes challenging.

In recent years, numerous day-ahead scheduling approaches have been proposed for this problem [155, 156, 157, 158, 159]. For instance, in order to handle the uncertainty in the charging demand, D. Wu et al. [155] developed a two-stage stochastic optimization method for workplace EV charging management. In a similar way, Y. Guo et al. [156] proposed a two-stage framework for the economical operation of a microgrid-like EV parking deck while taking the intermittency of renewable outputs into account. I. Momber et al. [157] developed a two-stage stochastic linear programming to maximize the EV aggregator's profit in both day-ahead and balancing market while the uncertainties of EV fleet mobility and market price were considered. In [158, 159], the EV aggregator was considered to bid in the day-ahead market and offered ancillary services. Stochastic programming approaches were applied to manage the EV fleet charging while taking into account the randomness of the regulation signals. Although the aforementioned methods achieved some success in day-ahead charging/discharging scheduling, they may be unsuitable for real-time scenarios where the variations in EV charging demand and

the electricity prices are much more complex.

Real-time scheduling strategies that can respond to dynamic charging demand and time-varying electricity prices have attracted a lot of attention recently. For example, L. Yao et al. [160] developed a binary programming-based strategy to coordinate multiple EVs charging in a parking station in response to real-time curtailment request from the utility. G. Binetti et al. [161] offered a formulation for the coordinated charging problem which considered the plug-in and plug-off frequency. Then, a real-time greedy algorithm is designed to solve the formulated problem in a decentralized manner. Y. T. Liao et al. [162] employed a stochastic dynamic programming method which maximizes the operator's profit for the real-time dispatch of an EV charging station equipped with photovoltaic panels. Q. Huang et al. [163] presented an MDP formulation for the EV scheduling problem while considering the uncertainty and dynamics in wind energy supply. A rollout algorithm is used to derive the optimal scheduling policy. The above methods formulate this scheduling problem as a model-based control problem which generally requires a model of forecast on the uncertainty and optimization for the scheduling process. The crucial step in constructing a model-based controller includes how to properly select the model and accurately estimate its parameters. This step becomes more challenging when the heterogeneity of the EV users, such as their commuting behavior, are taken into consideration. Consequently, different model parameters and even different models are required for different users.

In order to solve this problem, in this chapter, a data-driven approach that does not need any system model information is proposed [62]. The EV charging management problem is formulated as an MDP from the user's perspective. The objective is to find cost-efficient charging schedules to take advantage of the real-time electricity price while fulfilling user's driving demand.

5.3 Problem Formulation

The real-time EV charging management problem is formulated from the user's perspective. When the EV is at home, the charging action is determined every hour. A finite MDP with discrete time step is applied to formulate this problem. Specifically, the time interval between two adjacent steps is one hour. At time step t , the system state s_t , which includes the information about the remaining energy in the EV battery and the past 24-hour electricity prices, is observed. Based on this information, the charging action a_t is selected. This action represents the amount of energy that the EV battery will be charged or discharged during this time interval. After executing this action, the system will proceed to the next state s_{t+1} . The details about the MDP formulation are shown as follows.

State

The system state at time step t is defined as a vector $s_t = (u_t, E_t, P_{t-23}, \dots, P_t)$. This vector encapsulates three types of information: (1) u_t indicates whether the EV is at home or not; (2) E_t represents the remaining energy in the EV battery; (3) $(P_{t-23}, \dots, P_t)$ denotes the past 24-hour electricity prices.

Action

Given the state s_t , the action a_t represents the charging/discharging power. When the EV is charging, a_t is positive. When the EV is discharging, a_t is negative. The EV user can make money by charging the battery when the electricity price is low and discharging the battery when the price is on-peak. The charging and discharging power is constrained as below,

$$-e_{dis}^{max} \leq a_t \leq e_{ch}^{max}, \quad (46)$$

where e_{ch}^{max} and e_{dis}^{max} are the allowed maximum charging and discharging power of the EV battery, respectively. The EV charger provides discrete charging power,

i.e., $a_t \in \{e^1, \dots, e^L\}$.

State transition

The state transition from the state s_t to s_{t+1} is denoted as

$$s_{t+1} = f(s_t, a_t, \omega_t), \quad (47)$$

where the state transition is not only controlled by the action a_t but also influenced by the randomness ω_t . Specifically, the state transition for E_t is controlled by a_t and can be explicitly expressed by a deterministic battery model $E_{t+1} = E_t + a_t$. For u_t and P_t , the state transition is subject to randomness since the arrival time, departure time, and next-hour electricity price are unknown. Finding an accurate distribution model of the randomness ω_t can be difficult since it is influenced by many factors, such as traffic condition, user's commuting behavior, pricing process of the utility, etc. To solve this problem, a data-driven approach is proposed to learn the state transition directly from the input data.

Reward

The reward at time step t is defined from user's perspective as

$$r_t = \begin{cases} -P_t \cdot a_t, & t \neq t_\beta \\ -P_t \cdot a_t - \tau \cdot (E_{max} - E_t)^2, & t = t_\beta \end{cases} \quad (48)$$

where t_β is the time step when the EV leaves home.

In this reward, $P_t \cdot a_t$ represents the charging cost at time step t . When the EV is charging, this term is positive. On the contrary, if the EV discharges energy back to the power grid to make money, this term is negative.

$(E_{max} - E_t)^2$ measures user's "range anxiety" which is the fear that the EV has insufficient energy to reach its destination. This term is proposed to penalize

the amount of uncharged battery energy, $E_{max} - E_t$. Larger uncharged energy will result in bigger penalty. In practice, some users are likely to tolerate large range anxiety to obtain low charging cost. In order to measure the user's preference towards the cost-saving objective and the range anxiety reducing objective, an anxiety coefficient τ is introduced. τ is measured in $\$/\text{kWh}^2$ such that the range anxiety term has the same measurement unit as the charging cost term.

Action-value function

The quality of a charging action a under a given system state s is assessed by the expected total sum of future rewards for the horizon of K time steps as follow

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{k=0}^K \gamma^k \cdot r_{t+k} \middle| s_t = s, a_t = a \right], \quad (49)$$

where $Q_{\pi}(s, a)$ is called action-value function, π is the EV charging policy which maps from a system state to a charging schedule, and $0 < \gamma < 1$ is the discount factor, which balances the importance between the immediate reward and future rewards. For instance, when $\gamma = 1$, future reward is as important as the immediate reward and the policy is foresighted. When $\gamma = 0$, only immediate reward is taken into consideration and the policy is shortsighted.

The objective of this scheduling problem is to find an optimal π^* to maximize the action-value function as

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a), \quad (50)$$

where $Q^*(s, a)$ represents the optimal action-value function.

5.4 Data-Driven EV Charging Management

It is difficult to analytically determine the optimal policy π^* since the future electricity prices and user's commuting behavior are unknown. A data-driven

solution is to iteratively update the action-value function $Q(s, a)$ based on the Bellman equation [164] as

$$Q_{i+1}(s, a) = \mathbb{E} \left[r_t + \gamma \max_{a_{t+1}} Q_i(s_{t+1}, a_{t+1}) \middle| s_t = s, a_t = a \right]. \quad (51)$$

As the number of iterations $i \rightarrow \infty$, $Q(s, a)$ will converge to the optimal action-value function $Q^*(s, a)$ [52]. Then, the optimal schedules are determined by a greedy strategy

$$a^* = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a). \quad (52)$$

In traditional works, $Q^*(s, a)$ is generally approximated by a look-up table [52]. However, since the electricity price in this problem is continuous and high-dimensional, an extremely large table is required to approximate $Q^*(s, a)$ and it is intractable to update such a large table. In the proposed data-driven approach, a DNN is used to approximate $Q^*(s, a)$.

The overall diagram of the proposed approach based on DRL for EV charging management is illustrated in Fig. 36. The representation network extracts discriminative features from the electricity price. After concatenating these features with battery SOC, the concatenated features are fed into a Q network to approximate the action-value of all feasible schedules under the given input state. The schedule with the largest action-value is selected as the EV charging schedule.

Extracting discriminative features from the raw data is a crucial step to improve the action-value function approximation. Good features should contain the information about the electricity price trends. With these features, the scheduling strategy can minimize the charging cost. In this paper, a representation network is proposed to extract these features.

Since electricity price fluctuates in a quasiperiodic way and has a natural

Figure 36: The Overall architecture of the proposed data-driven EV charging management.

temporal ordering, it is reasonable to infer future price trends from past electricity prices. LSTM network is known for its strong ability to model the time dependencies of time-series data [165], and has achieved promising results in smart grid applications, such as load forecasting [166, 167]. In the representation network of Fig. 36, electricity price trends are captured by a LSTM network. Its input is the past 24-h electricity price and its output is the features containing information about future price trends.

The output of the LSTM network, y_t , is concatenated with the battery SOC which is a scalar. These concatenated features, x_t , contain information about both the future price trends and the battery SOC. The information of the future price trends is essential to reduce the charging cost, while the information of the battery SOC is important to ensure the EV can be well-charged. Then, these concatenated features are fed into the Q network to approximate the optimal action-value function.

The Q network is a three-layer fully-connected neural network. The input layer is fully-connected to a hidden layer with V units, and the value of the hidden

unit is

$$v_t = g(W_1 * x_t + b_1), \quad (53)$$

where x_t denotes the input vector, g is the rectified linear activation function, W_1 is the matrix of weights, and b_1 is the vector of biases. Then, the hidden layer is fully-connected to the output layer. The output of the Q network is the action-value for all feasible charging/discharging schedules under system state s , i.e.,

$$Q(s, a) = g(W_2 * v_t + b_2), \quad (54)$$

where W_2 is the matrix of weights, and b_2 is the vector of biases. Then, the schedule with the largest action-value is outputted as the charging schedule.

Algorithm 3 shows how to train the DNN based on RL. The parameters of the DNN are denoted as $\theta = \{\theta_1, \theta_2\}$, where θ_1 represent the parameters of the representation network, and θ_2 represent the parameters of the Q network. θ_1 and θ_2 are shown in Eq. (55) and (56), respectively.

$$\theta_1 = \{W_z, R_z, b_z, W_i, R_i, b_i, W_f, R_f, b_f, W_o, R_o, b_o\} \quad (55)$$

$$\theta_2 = \{W_1, b_1, W_2, b_2\} \quad (56)$$

The input of Algorithm 3 is the electricity prices, battery SOC, and reward r . Its output is the DNN's parameters θ that include the parameters of the representation network and the parameters of the Q network.

The DNN's parameters trained by Algorithm 3 will be fixed for the real-time EV charging management. The scheduling algorithm is presented in Algorithm 4. Its input is the electricity prices and battery SOC. Its output is the EV charging schedules.

Algorithm 3 Training of Deep Neural Network

Input: Electricity prices, battery SOC, and reward r

Output: DNN's parameters θ

- 1: Randomly initialize DNN's parameters θ .
 - 2: Initialize shadow parameters $\bar{\theta} = \theta$.
 - 3: **for** Epoch=1:M **do**
 - 4: Obtain the initial state s_{t_A} .
 - 5: **for** Time step $t=t_A:t_D$ **do**
 - 6: Select schedule a_t based on ε -greedy search.
 - 7: Execute schedule a_t , observe reward r_t , and process to the new state s_{t+1} .
 - 8: Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$.
 - 9: Sample minibatch of transitions $\mathcal{F} = \{(s_j, a_j, r_j, s_{j+1})\}_{j=1}^{\#\mathcal{F}}$ from $\mathcal{D}$.
 - 10: $\bar{q}_j \leftarrow r_j + \gamma Q(s_{j+1}, \arg\max_a Q(s_{j+1}, a; \theta_t); \bar{\theta})$.
 - 11: Calculate the loss function $L(\theta_t) = \sum_{j=1}^{\#\mathcal{F}} [\bar{q}_j - Q(s_j, a_j; \theta_t)]^2$.
 - 12: Update parameters $\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L(\theta_t)$.
 - 13: Every B steps reset $\bar{\theta} = \theta$.
 - 14: **end for**
 - 15: **end for**
-

Algorithm 4 first loads the parameters of DNN trained by Algorithm 3. In the loop starting from line 2, the DNN is implemented to generate the EV charging schedules from time step t_A to t_D . At each time step, the representation network is applied to extract features from the past 24-h electricity prices. Then, these features are concatenated with battery SOC. After that, Q network calculates action-value $Q(s_t, a; \theta)$ based on these concatenated features. Then, the EV charging schedule a_t is selected in line 7 as $a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a; \theta)$. Finally, the schedule a_t is outputted.

5.5 Experimental Results

In this section, the proposed data-driven EV charging management is evaluated under simulation studies. This section will first introduce the details of the experimental setup. Then, this section will present the experimental results on a real-world scenario. Then, this section will investigate how to balance the cost-

Algorithm 4 Real-Time EV Charging Management

Input: State s_t , including electricity prices and battery SOC.

Output: EV charging/discharging schedules $a_{t_A:t_D}$.

- 1: Load the DNN's parameters θ trained by Algorithm 3.
 - 2: **for** Time step $t=t_A:t_D$ **do**
 - 3: Obtain past 24-h electricity prices.
 - 4: Representation network extracts features from the electricity prices.
 - 5: Concatenate these features with battery SOC.
 - 6: Q network calculates action-value $Q(s_t, a; \theta)$.
 - 7: $a_t = \operatorname{argmax}_{a \in \mathcal{A}} Q(s_t, a; \theta)$
 - 8: Output EV charging/discharging schedule a_t .
 - 9: **end for**
 - 10: **Return:** $a_{t_A:t_D}$
-

saving objective and the range anxiety reducing objective. Finally, this section will conduct ablation studies to explain how to choose the neural network structures and the hyperparameters.

5.5.1 Experimental Setup

The performance of the proposed data-driven approach is evaluated under a real-world scenario. The real-world hourly electricity price starting from February 1st, 2014 and lasting 360 days is downloaded from the California ISO [168]. This price data is separated into training and testing data. In each 30 consecutive days, the first 20 days are selected for training, and the remaining 10 days are used for performance evaluation. User's commuting behavior is modeled as random variables. Specifically, the EV arrival time, the departure time, and the battery SOC at the arrival time obey truncated normal distributions. The distributions are presented in Table. 14. The arrival time t_A is sampled from $\mathcal{N}(18, 1^2)$ and is bounded between 15 and 21. For the departure time t_D , its distribution $\mathcal{N}(8, 1^2)$ is bounded between 6 and 11. The battery SOC bounded between 0.2 and 0.8 is sampled from $\mathcal{N}(0.5, 0.1^2)$. It is worth noting that the proposed data-driven approach does not rely on any knowledge of the distributions of these random

variables. Thus, it can scale to different modeling mechanisms. A Nissan Leaf with battery capacity $E_{max} = 24$ kWh is used in the experiments. When the EV is at home, the charger provides 7 levels (6 kW, 4 kW, 2 kW, 0 kW, -2 kW, -4 kW, -6 kW) for EV charging and discharging. The positive values represent the charging process while the negative ones refer to EV discharging.

Table 14: Random variables for commuting behavior.

	Distribution	Boundary
Arrival time	$t_A \sim \mathcal{N}(18, 1^2)$	$15 \leq t_A \leq 21$
Departure time	$t_D \sim \mathcal{N}(8, 1^2)$	$6 \leq t_D \leq 11$
Battery SOC	$SOC \sim \mathcal{N}(0.5, 0.1^2)$	$0.2 \leq SOC \leq 0.8$

The DRL based data-driven approach is proposed to generate the optimal charging schedules. The discounted factor γ is set to 0.99 so that the proposed approach can obtain a foresighted strategy. In the representation network, a 128-dimension feature vector is extracted from the past 24-h electricity price. Then, this vector is concatenated with the SOC and fed into the input layer of the Q network. The number of the units in its hidden layer and output layer are 64 and 7, respectively. The parameters of the representation network and the Q network are randomly initialized and updated by gradient descent during the training process. The batch size of the sampled transitions $\mathcal{F}$ for training is 32. The value of B is set as 500. The number of training epochs, M , equals to 50,000. The training process takes about 1.5h on the computer with one NVIDIA TITAN Xp GPU and one i7-6800K CPU. After the training process, the proposed approach can be deployed for the EV charging management. It takes about 3.7 ms to generate one schedule. The code is written in Python with TensorFlow.

5.5.2 Performance Evaluation

The proposed DNN is trained for 50,000 epochs to learn the optimal EV charging management policy. The anxiety coefficient τ is set to 0.01. The training process takes about 1.5 hours. Each epoch starts when the EV arrives home and ends when it leaves home. In each epoch, the cumulative rewards $\sum_{t=t_A}^{t_D} r_t$ is calculated. The evolution of the cumulative rewards over 50,000 epochs is illustrated in Fig. 37. In the first 7,000 epochs, the charging/discharging schedule is randomly selected from the 7 feasible levels. Then, from epoch 7,000 to epoch 14,000, the schedule is randomly selected with probability ε , otherwise, it is chosen by the proposed DNN. In this phase, the probability ε is reduced from 1.0 to 0.1, and remains 0.1 afterward. It can be observed from Fig. 37 that the cumulative rewards start to increase gradually after epoch 18,000. Then, at epoch 35,000, the cumulative rewards converge around -0.2 \$ with small oscillations. This result demonstrates that the proposed approach succeeds in learning a policy to maximize the cumulative rewards.

Figure 37: The evolution of the cumulative rewards during the training process.

The proposed approach is evaluated and compared with several benchmark solutions, including model predictive control (MPC), day-ahead, fitted-Q iteration (FQI), and uncontrolled strategy. For the MPC solution, the EV arrival time and depart time, and the battery SOC are known in advance. MPC is evaluated with two different forecasting models. For the first forecasting model, a fully-connected neural network (NN) with 24-20-20-8 units is used to provide the dynamic prediction of the future electricity prices over a rolling horizon of 8 hours. Based on the forecasted prices, a scheduling strategy over the rolling horizon is derived and only the first hour's schedule is implemented. At the next hour, the above procedure is repeated. Similarly, for the second forecasting model, an LSTM network is used to forecast the future electricity prices. This LSTM network is the same as the one in the representation network. MPC is also evaluated under an **ideal** situation where the future electricity prices are all known in advance. This ideal solution provides an upper bound for the performance. However, it cannot be reached in reality because of the uncertainty in the future prices. For the day-ahead solution, the EV arrival time and depart time, and the battery SOC are known in advance. Autoregressive (AR) model is applied to forecast the electricity prices, and its order is 24, i.e. AR (p=24). Then, the EV charging management problem is solved using YALMIP. For the FQI solution [169], a decision tree is used to approximate the action-value function. Then, the charging schedule is determined based on the greedy strategy. For the uncontrolled solution, the EV is charged immediately with the maximum charging rate when it arrives home.

The 120 test days are used for performance evaluation. In each day, the charging cost is calculated as $\sum_{t=t_A}^{t_D} P_t \cdot a_t$. The cumulative charging costs of the proposed and benchmark solutions over the test days are presented in Fig. 38. The percentage terms on the right illustrate the cost reduction of the corresponding solutions

with respect to the uncontrolled solution. We can observe that the proposed approach (blue solid line) reduces the charging cost by 77.30% in comparison with the uncontrolled solution (purple dotted line). The other four benchmark solutions, FQI (orange dash-dotted line), day-ahead (black circles), MPC+NN (red dash line), and MPC+LSTM (black dotted line) only reduce the cost by 46.29%, 59.10%, 69.87%, and 75.04%, respectively. These results demonstrate the effectiveness of the proposed approach for this EV charging management problem. We also notice that MPC with real future price (green triangles) reduces the charging cost by 86.29%. Comparing the results of MPC, we can find that the performance of MPC relies on the accuracy of a forecasting model. However, the proposed data-driven approach does not need a forecasting model.

Figure 38: Cumulative charging cost of the proposed and benchmark solutions over the 120 test days.

To further investigate the performance of the proposed approach, the charging/discharging patterns over 7 consecutive days are shown in Fig. 39. The green

(a) Hourly electricity price (red line) and charging/discharging energy (blue bar).

(b) Remaining battery energy

Figure 39: Charging/discharging patterns of the proposed approach over 7 consecutive days.

regions in each subfigure indicate the periods of time when the EV is not at home. In Fig. 39a, the hourly electricity price (\$/kWh) is illustrated with the red line, while the hourly charging/discharging energy (kWh) is represented by the blue bar. One can observe that the proposed approach learns to charge when the electricity price is low and to discharge when the price is on-peak. These charging/discharging patterns verify the proposed approach's capability to forecast the electricity price trends. The remaining battery energy at each hour is presented in Fig. 39b. The battery is well-charged when the EV leaves home. These results demonstrate that the proposed approach can reduce the charging cost as well as satisfy user's driving

demand.

Figure 40: Average range anxiety (blue circle) and charging cost (red square) under different anxiety coefficient.

In the real-world scenario, different users may have a different preference towards the cost-saving objective and the range anxiety reducing objective. The impact of the anxiety coefficient τ on these two objectives is demonstrated in Fig. 40 where the coefficient increases from 0 to 0.01 with a step of 0.001. In Fig. 40, the range anxiety and charging cost are averaged over the 120 test days. The blue line with circle marks represents the average range anxiety, while the red line with square marks is the curve for average charging cost. The negative value of the charging cost indicates that the EV earns money by selling electricity to the utility. We can observe that large anxiety coefficient leads to small range anxiety but large charging cost. This figure demonstrates that the coefficient can balance the trade-off between the range anxiety and the charging cost. Specifically, if the user only needs to drive for a short distance, the coefficient can be set to a small

value. Consequently, the proposed approach tries to reduce the charging cost. On the contrary, if the user wants to reduce the range anxiety, this coefficient can be set to a high value.

In order to further demonstrate the effect of the anxiety coefficient τ , the daily charging cost and the battery SOC when the EV leaves home are presented in Fig. 41. This figure shows the results over 120 test days with two different τ . When τ is set to 0.002, the proposed approach can achieve a lower charging cost, but the battery is about half charged. On the contrary, when τ is set to 0.008, the battery is almost fully charged, but the charging cost increases. These results demonstrate that the proposed approach can adaptively adjust to user's different preferences by setting different anxiety coefficient.

5.5.3 Ablation Studies

The reason LSTM network is used to extract features from the electricity prices is that it has strong ability to model the time dependencies of time-series data [165]. To investigate the effectiveness of the LSTM network, an ablation experiment is conducted by removing the LSTM network. Specifically, the electricity prices are directly concatenated with the battery SOC. Then, these concatenated features are fed into the Q network. Without the LSTM network, the accumulated charging cost over the 120 test days is 27.71 \$. However, the accumulated charging cost of the original model with the LSTM network is only 17.02 \$. The LSTM network contributes to a 38.59% reduction in the charging cost.

For the fully-connected neural network, there is an empirically-derived rule-of-thumb to choose the number of hidden units. The optimal number of the hidden units is usually between the number of input units and the number of the output units. In the proposed model, the number of hidden units is 64, and the accumulated charging cost over the 120 test days is 17.02 \$. When the number of

(a) Charging cost

(b) Battery SOC

Figure 41: Daily charging cost and battery SOC when the EV leaves home under different anxiety coefficient τ .

hidden units is chosen as 32 and 128, their accumulated charging cost are 17.29 \$ and 17.13 \$, respectively. We can see that the performance is insensitive to the number of hidden units when this number is chosen between the number of input units and the number of the output units.

The discounted factor γ balances the importance between the immediate reward and future rewards. When $\gamma = 0$, the learned policy is “myopic” in being concerned only with maximizing immediate reward. As γ approaches 1, the future rewards are taken into consideration more strongly, and the learned policy becomes more farsighted. In the original model, $\gamma = 0.99$, which is the same as that used in [50]. For the original model, the accumulated charging cost over the 120 test days is 17.02 \$. When γ is chosen as 0.5 and 0.8, their accumulated charging cost are 36.92 \$ and 24.65 \$, respectively. We can observe that the performance is sensitive to the discounted factor γ . γ should be close to 1 such that the learned policy is farsighted.

5.6 Chapter Summary

This chapter proposed a data-driven approach to determine an optimal strategy for the real-time EV charging management problem. This problem was formulated as an MDP with unknown transition probability from the user’s perspective. In the problem formulation, the randomness of both the electricity price and the commuting behavior were considered. The proposed approach does not need any system model information. It can directly learn the optimal policy from the input data. In the proposed approach, a representation network is designed to extract features from the electricity prices. Then, a Q network is implemented to approximate the optimal action-value function based on these features. Finally, the action that maximizes this function is selected for the scheduling problem. The parameters of these two networks are updated by gradient descent during the training

process. Experimental results showed that the proposed approach outperforms various benchmark solutions for this EV charging problem. In addition, the proposed approach can adaptively adjust to different users' preferences towards the cost-saving objective and the range anxiety reducing objective. Finally, ablation experiments were conducted to investigate how to choose the neural network structure and the hyperparameters. The proposed approach can also be extended to many other applications in smart grid, such as residential energy management [170, 171, 172] and cyber-physical attack and recovery [173, 174].

CHAPTER 6

Conclusions

6.1 Summary of the Dissertation

This dissertation first focuses on automatic feature extraction, which can avoid the burden of the “feature engineering” process. The domain shift problem for supervised feature extraction was investigated. A low-level feature alignment method was proposed to reduce the domain shift. The experimental results on adapting from clear weather to foggy weather and adapting from synthetic images to real-world images showed that the proposed domain adaptation method can maintain the performance of the feature extractor across different domains. The ablation results showed that the proposed low-level feature alignment is much more effective in domain shift reduction than high-level feature alignment. In addition, the proposed domain adaptation method is insensitive to the choice of the target sample. This dissertation also studied an unsupervised feature extractor, sparse autoencoder. It does not need any labeling information and therefore does not suffer from the domain shift problem. A generative model was established for the sparse autoencoder to determine the sparse hyperparameter. Unlike the traditional works that use brute-force search to determine this hyperparameter, the proposed approach is much more efficient. Experimental results on MNIST dataset validated the effectiveness of the proposed approach in determining the sparse hyperparameter.

Then, this dissertation studied the heterogeneous data fusion problem. Two important types of data, image and natural language, were considered. An attention network was used to calculate the correlation between the input data. The image features were refined according to the correlation to the question. To effectively fuse the refined images features and question features, a hierarchical fusion

network was developed. In this fusion network, the image features and question features were transformed into a common feature space. Then, a multi-head fusion was proposed to fuse these transformed features at multiple levels. The experimental results on VQA dataset showed that the proposed heterogeneous data fusion outperformed other benchmark solutions.

With the automatic feature extraction and heterogeneous data fusion, this dissertation developed a data-driven decision-making framework for robot-assisted pedestrian regulation. In this pedestrian regulation problem, a mobile robot was introduced to regulate the pedestrian flows through passive HRI. A data-driven approach based on DRL was proposed to generate the robot control actions only with the raw image observation. The experimental results showed that the proposed approach succeeded in learning optimal robot motion planning strategy to maximize the accumulated outflow under different inflow ratios and different robot initial positions. In addition, the visualization results of the extracted features revealed how the automatic feature extraction contributed to the success of the proposed data-driven approach.

This dissertation also proposed a data-driven approach for another real-world problem, real-time EV charging management. This problem was formulated as an MDP where the randomness of both the electricity prices and user’s commuting behavior were considered. The proposed approach learned the optimal charging policy in a “model-free” fashion, which means that it does not need any system model information. In the proposed approach, an LSTM network was utilized to extract the future price trends from the past electricity prices. Then, a Q network was implemented to approximate the optimal action-value function based on these features. Experimental results under real-world scenarios showed that the proposed approach can significantly reduce the charging cost and outperformed

various benchmark solutions. In addition, experimental results showed that the proposed approach can well adapt to different users who may have different preferences towards the cost-saving and range anxiety reducing.

6.2 Challenges and Opportunities

There are significant research directions and future opportunities following the work presented in this dissertation. This dissertation would like to highlight the following three critical future directions.

Explainable Data-Driven Decision-Making

Due to the black-box nature of DNN, it is extremely hard for humans to understand how the data-driven system derives such decision. In many critical domains, such as health care, legislation, and law enforcement, it is of paramount importance to make sure the decisions are legally or ethically defensible [175]. In addition, the General Data Protection Regulation (GDPR) requires the automated decision-making systems to provide an explanation of how the decision is reached [176]. These requirements call for novel theories and tools to explain and understand the functionality of DNN. One potential way is to investigate the learning mechanism to derive principles that can explain the evolution of DNN's parameters during the training process. In addition, semantic concepts can be assigned to DNN's weights or feature representations such that we can uncover a human-interpretable explanation of the learning mechanism. For example, in CNN, these semantic concepts can be generated based on the visualization of the convolutional kernels and feature maps.

Secure Data-Driven Decision-Making

Another future direction is to enhance the security of the data-driven decision-making systems. For real-world systems, such as power system, the measurement

units and the control center are being increasingly interconnected via communication facilities, which makes these systems exposed to numerous cyber-attacks. These attacks can undermine the system performance and, sometimes, may cause severe damage to the system. A real-world incident is the cyber-attack on Ukraine power substation in 2015 [177]. During this attack, 30 substations were shut down, and over 225,000 people lost power. For DNN-based data-driven systems, we should also study the vulnerability to a special type of attack, adversarial attack. During adversarial attack, the hacker can attack the output of DNN by intentionally injecting small amount of perturbation into the input data [178]. It is crucial to understand the vulnerability of the data-driven systems to different types of attacks. In addition, advanced defense strategies should be developed to make the data-driven systems resilient and robust to these attacks.

Computationally Efficient Data-Driven Decision-Making

The existing DNNs are computationally expensive and memory intensive, which hinders the deployment of data-driven systems in applications with strict latency requirements or in devices with low memory resources [179]. A potential solution is to compress and accelerate the DNN without significantly undermining the model performance. In DNN, there exists some redundant parameters. We can find and prune these parameters according to the sensitivity to the performance. DNN can be further compressed by reducing the number of bits used to represent each parameter.

LIST OF REFERENCES

- [1] M. Wall. “Big data: Are you ready for blast-off?” Accessed on June 2, 2020. [Online]. Available: <https://www.bbc.com/news/business-26383058>
- [2] C. P. Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on big data,” *Information Sciences*, vol. 275, pp. 314 – 347, 2014.
- [3] J. Lu, Z. Yan, J. Han, and G. Zhang, “Data-driven decision-making (d3m): Framework, methodology, and directions,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 4, pp. 286–296, 2019.
- [4] H. Wang, Z. Xu, H. Fujita, and S. Liu, “Towards felicitous decision making: An overview on challenges and trends of big data,” *Information Sciences*, vol. 367-368, pp. 747 – 765, 2016.
- [5] E. Brynjolfsson and K. McElheran, “The rapid adoption of data-driven decision-making,” *American Economic Review*, vol. 106, no. 5, pp. 133–39, May 2016.
- [6] G. Cao, Y. Duan, and G. Li, “Linking business analytics to decision making effectiveness: A path model analysis,” *IEEE Transactions on Engineering Management*, vol. 62, no. 3, pp. 384–395, 2015.
- [7] W. Kellerer, P. Kalmbach, A. Blenk, A. Basta, M. Reisslein, and S. Schmid, “Adaptable and data-driven softwarized networks: Review, opportunities, and challenges,” *Proceedings of the IEEE*, vol. 107, no. 4, pp. 711–731, 2019.
- [8] T. Carriere and G. Kariniotakis, “An integrated approach for value-oriented energy forecasting and data-driven decision-making application to renewable energy trading,” *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6933–6944, 2019.
- [9] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.
- [10] M. Nixon and A. Aguado, *Feature extraction and image processing for computer vision*. Academic press, 2019.
- [11] H. Liu and H. Motoda, *Feature extraction, construction and selection: A data mining perspective*. Springer Science & Business Media, 1998, vol. 453.
- [12] L. Wang, “Heterogeneous data and big data analytics,” *Automatic Control and Information Sciences*, vol. 3, no. 1, pp. 8–15, 2017.

- [13] Y. Wu, E. Y. Chang, K. C.-C. Chang, and J. R. Smith, "Optimal multimodal fusion for multimedia data analysis," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, 2004, pp. 572–579.
- [14] S. Poria, E. Cambria, A. Hussain, and G.-B. Huang, "Towards an intelligent framework for multimodal affective data analysis," *Neural Networks*, vol. 63, pp. 104–116, 2015.
- [15] D. Che, M. Safran, and Z. Peng, "From big data to big data mining: Challenges, issues, and opportunities." Springer Berlin Heidelberg, 2013.
- [16] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.
- [17] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, "Big data: Issues and challenges moving forward," in *2013 Hawaii International Conference on System Sciences*, 2013, pp. 995–1004.
- [18] N. Khan, I. Yaqoob, I. A. T. Hashem, Z. Inayat, W. K. Mahmoud Ali, M. Alam, M. Shiraz, and A. Gani, "Big data: survey, technologies, opportunities, and challenges," *The Scientific World Journal*, 2014.
- [19] D. G. Lowe, "Object recognition from local scale-invariant features," in *1999 IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.
- [20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.
- [21] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.
- [22] B. Jan, H. Farman, M. Khan, M. Imran, I. U. Islam, A. Ahmad, S. Ali, and G. Jeon, "Deep learning in big data analytics: A comparative study," *Computers & Electrical Engineering*, vol. 75, pp. 275–287, 2019.
- [23] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, 2015.
- [24] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.
- [25] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

- [26] Y. Bengio and O. Delalleau, “On the expressive power of deep architectures,” in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 18–36.
- [27] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.
- [28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.
- [29] L. Wu, Y. Wang, X. Li, and J. Gao, “Deep attention-based spatially recursive networks for fine-grained visual recognition,” *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1791–1802, 2019.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, Sept 2015.
- [31] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.
- [32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.
- [33] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *2015 International Conference on Learning Representations (ICLR)*, 2015.
- [34] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 649–657.
- [35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.
- [36] 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*, 2018.
- [37] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

- [38] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, *et al.*, “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *International Conference on Machine Learning*, 2016, pp. 173–182.
- [39] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.
- [40] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.
- [41] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.
- [42] D. H. Hubel and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex,” *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.
- [43] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
- [44] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.
- [46] F. Li, J. Niebles, and S. Savarese. “CS231n: Convolutional neural networks for visual recognition.” Accessed on September 21, 2020. [Online]. Available: <https://cs231n.github.io/convolutional-networks/>
- [47] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [48] D. Lee, M. L. Conroy, B. P. McGreevy, and D. J. Barraclough, “Reinforcement learning and decision making in monkeys during a competitive game,” *Cognitive Brain Research*, vol. 22, no. 1, pp. 45–58, 2004.
- [49] E. O. Neftci and B. B. Averbeck, “Reinforcement learning in artificial and biological systems,” *Nature Machine Intelligence*, vol. 1, no. 3, pp. 133–143, 2019.

- [50] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [51] M. Wandel. “Bigger, tighter mouse maze experiments.” Accessed on January 12, 2020. [Online]. Available: https://woodgears.ca/farm/mouse_maze.html
- [52] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
- [53] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- [54] M. Van Otterlo and M. Wiering, “Reinforcement learning and markov decision processes,” in *Reinforcement Learning*. Springer, 2012, pp. 3–42.
- [55] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [56] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, “Deepstack: Expert-level artificial intelligence in heads-up no-limit poker,” *Science*, vol. 356, no. 6337, pp. 508–513, 2017.
- [57] Deepmind. “Alphastar: Mastering the real-time strategy game starcraft II.” Accessed on September 22, 2020. [Online]. Available: <https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii>
- [58] Z. Wan, L. Li, H. Li, H. He, and Z. Ni, “One-shot unsupervised domain adaptation for object detection,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.
- [59] Z. Wan, H. He, and B. Tang, “A generative model for sparse hyperparameter determination,” *IEEE Transactions on Big Data*, vol. 4, no. 1, pp. 2–10, 2018.
- [60] Z. Wan and H. He, “Answernet: Learning to answer questions,” *IEEE Transactions on Big Data*, vol. 5, no. 4, pp. 540–549, 2019.
- [61] Z. Wan, C. Jiang, M. Fahad, Z. Ni, Y. Guo, and H. He, “Robot-assisted pedestrian regulation based on deep reinforcement learning,” *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1669–1682, 2020.

- [62] Z. Wan, H. Li, H. He, and D. Prokhorov, “Model-free real-time EV charging scheduling based on deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5246–5257, 2019.
- [63] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.
- [64] Z. Wan and H. He, “Weakly supervised object localization with deep convolutional neural network based on spatial pyramid saliency map,” in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 4177–4181.
- [65] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.
- [66] J. Zhang, Z. Ding, W. Li, and P. Ogunbona, “Importance weighted adversarial nets for partial domain adaptation,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8156–8164.
- [67] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, “Universal domain adaptation,” in *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2720–2729.
- [68] J. Wen, N. Zheng, J. Yuan, Z. Gong, and C. Chen, “Bayesian uncertainty matching for unsupervised domain adaptation,” *arXiv:1906.09693*, 2019.
- [69] Y. Shu, Z. Cao, M. Long, and J. Wang, “Transferable curriculum for weakly-supervised domain adaptation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4951–4958.
- [70] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, “Domain adaptation via transfer component analysis,” *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, Feb 2011.
- [71] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 136–144.
- [72] P. O. Pinheiro, “Unsupervised domain adaptation with similarity learning,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8004–8013.
- [73] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2962–2971.

- [74] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 818–833.
- [75] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.
- [76] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.
- [77] C. Sakaridis, D. Dai, and L. Van Gool, “Semantic foggy scene understanding with synthetic data,” *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, Sep 2018.
- [78] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, “Domain adaptive Faster R-CNN for object detection in the wild,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 3339–3348.
- [79] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1–8.
- [80] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 37–49.
- [81] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi, “Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 1, pp. 119–130, Jan 2016.
- [82] M. Shu and A. Fyshe, “Sparse autoencoders for word decoding from magnetoencephalography,” in *Proceedings of the third NIPS Workshop on Machine Learning and Interpretation in NeuroImaging*, 2013, pp. 1–7.
- [83] A. L. Barth and J. F. Poulet, “Experimental evidence for sparse firing in the neocortex,” *Trends in Neurosciences*, vol. 35, no. 6, pp. 345–355, 2012.
- [84] O. J. Hulme, M. Skov, M. J. Chadwick, H. R. Siebner, and T. Z. Ramsøy, “Sparse encoding of automatic visual association in hippocampal networks,” *NeuroImage*, vol. 102, pp. 458–464, 2014.

- [85] D. J. Field, “What is the goal of sensory coding?” *Neural Computation*, vol. 6, no. 4, pp. 559–601, 1994.
- [86] D. Field, “Scale-invariance and self-similar ‘wavelet’ transforms: an analysis of natural scenes and mammalian visual systems,” in *Wavelets, Fractals and Fourier Transforms: New Developments and New Applications*, Oxford University Press, pp. 151–193, 1993.
- [87] C. Poultney, S. Chopra, Y. L. Cun, *et al.*, “Efficient learning of sparse representations with an energy-based model,” in *Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 1137–1144.
- [88] Q. V. Le, “Building high-level features using large scale unsupervised learning,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 8595–8598.
- [89] K. Chen, J. Hu, and J. He, “A framework for automatically extracting over-voltage features based on sparse autoencoder,” *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 594–604, 2018.
- [90] C. Tao, H. Pan, Y. Li, and Z. Zou, “Unsupervised spectral spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, Dec 2015.
- [91] K. Sun, S. Huang, D. S. Wong, and S. Jang, “Design and application of a variable selection method for multilayer perceptron neural network with LASSO,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1386–1396, 2017.
- [92] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, “A multiobjective sparse feature learning model for deep neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3263–3277, Dec 2015.
- [93] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.
- [94] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.
- [95] K. Deb, “Multi-objective optimization,” in *Search Methodologies*. Springer, 2014, pp. 403–449.
- [96] M. Malinowski, M. Rohrbach, and M. Fritz, “Ask your neurons: A neural-based approach to answering questions about images,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1–9.

- [97] M. Ren, R. Kiros, and R. Zemel, “Exploring models and data for image question answering,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2953–2961.
- [98] Q. Wu, P. Wang, C. Shen, A. Dick, and A. v. d. Hengel, “Ask me anything: Free-form visual question answering based on knowledge from external sources,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4622–4630.
- [99] K. Chen, J. Wang, L.-C. Chen, H. Gao, W. Xu, and R. Nevatia, “ABC-CNN: An attention based convolutional neural network for visual question answering,” *arXiv:1511.05960*, 2015.
- [100] H. Xu and K. Saenko, “Ask, attend and answer: Exploring question-guided spatial attention for visual question answering,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 451–466.
- [101] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, “Stacked attention networks for image question answering,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 21–29.
- [102] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, “Neural module networks,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 39–48.
- [103] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, “Learning to compose neural networks for question answering,” in *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2016, pp. 1545–1554.
- [104] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, “VQA: Visual question answering,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2425–2433.
- [105] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.
- [106] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv:1207.0580*, 2012.
- [107] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus, “Simple baseline for visual question answering,” *arXiv:1512.02167*, 2015.
- [108] H. Noh, P. H. Seo, and B. Han, “Image question answering using convolutional neural network with dynamic parameter prediction,” in *2016 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 30–38.

- [109] M. Malinowski, M. Rohrbach, and M. Fritz, “Ask your neurons: A deep learning approach to visual question answering,” *International Journal of Computer Vision*, vol. 125, no. 1-3, pp. 110–135, 2017.
- [110] J. Lu, J. Yang, D. Batra, and D. Parikh, “Hierarchical question-image co-attention for visual question answering,” in *Advances In Neural Information Processing Systems (NIPS)*, 2016, pp. 289–297.
- [111] H. Noh and B. Han, “Training recurrent answering units with joint loss minimization for vqa,” *arXiv:1606.03647*, 2016.
- [112] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, “Multimodal compact bilinear pooling for visual question answering and visual grounding,” in *Conference on Empirical Methods in Natural Language Processing*. ACL, 2016, pp. 457–468.
- [113] H. Nam, J. Ha, and J. Kim, “Dual attention networks for multimodal reasoning and matching,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2156–2164.
- [114] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, “Making the V in VQA matter: Elevating the role of image understanding in visual question answering,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6325–6334.
- [115] P. B. Luh, C. T. Wilkie, S.-C. Chang, K. L. Marsh, and N. Olderman, “Modeling and optimization of building emergency evacuation considering blocking effects on crowd movement,” *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 687–700, 2012.
- [116] E. Boukas, I. Kostavelis, A. Gasteratos, and G. C. Sirakoulis, “Robot guided crowd evacuation,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 739–751, 2015.
- [117] B. Tang, C. Jiang, H. He, and Y. Guo, “Human mobility modeling for robot-assisted evacuation in complex indoor environments,” *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 5, pp. 694–707, 2016.
- [118] D. Helbing, L. Buzna, A. Johansson, and T. Werner, “Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions,” *Transportation science*, vol. 39, no. 1, pp. 1–24, 2005.
- [119] G. Frank and C. Dorso, “Room evacuation in the presence of an obstacle,” *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 11, pp. 2135–2145, 2011.

- [120] Y. Zhao, M. Li, X. Lu, L. Tian, Z. Yu, K. Huang, Y. Wang, and T. Li, "Optimal layout design of obstacles for panic evacuation using differential evolution," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 175–194, 2017.
- [121] K. Yamamoto and M. Okada, "Control of swarm behavior in crossing pedestrians based on temporal/spatial frequencies," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 1036–1048, 2013.
- [122] B. D. Eldridge and A. A. Maciejewski, "Using genetic algorithms to optimize social robot behavior for improved pedestrian flow," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, 2005, pp. 524–529.
- [123] J. A. Kirkland and A. A. Maciejewski, "A simulation of attempts to influence crowd dynamics," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, 2003, pp. 4328–4333.
- [124] C. Jiang, Z. Ni, Y. Guo, and H. He, "Learning human-robot interaction for robot-assisted pedestrian flow optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.
- [125] Z. Wan, X. Hu, H. He, and Y. Guo, "A learning based approach for social force model parameter estimation," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 4058–4064.
- [126] H. R. Beom and H. S. Cho, "A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 464–477, Mar 1995.
- [127] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Machine Learning*, vol. 23, no. 2, pp. 279–303, May 1996.
- [128] C. Gaskett, L. Fletcher, and A. Zelinsky, "Reinforcement learning for a vision based mobile robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2000, pp. 403–409.
- [129] Y. Duan, B. Cui, and H. Yang, "Robot navigation based on fuzzy RL algorithm," in *Advances in Neural Networks - ISNN 2008*, 2008, pp. 391–399.
- [130] H. Mhaskar, Q. Liao, and T. A. Poggio, "When and why are deep networks better than shallow ones?" in *Association for the Advancement of Artificial Intelligence*, 2017, pp. 2343–2349.
- [131] Z. Zhang, D. Zhao, J. Gao, D. Wang, and Y. Dai, "FMRQ-a multiagent reinforcement learning algorithm for fully cooperative tasks," *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1367–1379, June 2017.

- [132] Q. Zhang and D. Zhao, “Data-based reinforcement learning for nonzero-sum games with unknown drift dynamics,” *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 2874–2885, 2019.
- [133] Y. He, N. Zhao, and H. Yin, “Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, Jan 2018.
- [134] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. M. Leung, and Y. Zhang, “Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, Nov 2017.
- [135] Y. F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 285–292.
- [136] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2017, pp. 31–36.
- [137] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, “Deep reinforcement learning with successor features for navigation across similar environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2017, pp. 2371–2378.
- [138] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3357–3364.
- [139] F. Martinez-Gil, M. Lozano, and F. Fernández, “Marl-ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups,” *Simulation Modelling Practice and Theory*, vol. 47, pp. 259 – 275, 2014.
- [140] L. Torrey, “Crowd simulation via multi-agent reinforcement learning,” in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010, pp. 89–94.
- [141] D. Helbing, A. Johansson, and H. Z. Al-Abideen, “Dynamics of crowd disasters: An empirical study,” *Physical review E*, vol. 75, no. 4, p. 046109, 2007.
- [142] A. Johansson, D. Helbing, H. Z. Al-Abideen, and S. Al-Bosta, “From crowd dynamics to crowd safety: a video-based analysis,” *Advances in Complex Systems*, vol. 11, no. 04, pp. 497–527, 2008.

- [143] Y. Tajima and T. Nagatani, “Clogging transition of pedestrian flow in t-shaped channel,” *Physica A: Statistical Mechanics and its Applications*, vol. 303, no. 1, pp. 239–250, 2002.
- [144] J. Zhang, W. Klingsch, A. Schadschneider, and A. Seyfried, “Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2011, no. 06, p. P06004, 2011.
- [145] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [146] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in *Association for the Advancement of Artificial Intelligence*, 2016, pp. 2094–2100.
- [147] G. Ferrer, A. Garrell, and A. Sanfeliu, “Robot companion: A social-force based approach with human awareness-navigation in crowded environments,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1688–1694.
- [148] D. Vasquez, B. Okal, and K. O. Arras, “Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 1341–1346.
- [149] Z. Chen, C. Jiang, and Y. Guo, “Pedestrian-robot interaction experiments in an exit corridor,” in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 29–34.
- [150] L. v. d. Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [151] D. Leistikow. “The egallon: How much cheaper is it to drive on electricity?” Accessed on August 16, 2017. [Online]. Available: <https://energy.gov/articles/egallon-how-much-cheaper-it-drive-electricity>
- [152] T. Namerikawa, N. Okubo, R. Sato, Y. Okawa, and M. Ono, “Real-time pricing mechanism for electricity market with built-in incentive for participation,” *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 2714–2724, Nov 2015.
- [153] Maigha and M. L. Crow, “Cost-constrained dynamic optimal electric vehicle charging,” *IEEE Transactions on Sustainable Energy*, vol. 8, no. 2, pp. 716–724, April 2017.

- [154] J. R. Pillai and B. Bak-Jensen, "Integration of vehicle-to-grid in the western danish power system," *IEEE Transactions on Sustainable Energy*, vol. 2, no. 1, pp. 12–19, Jan 2011.
- [155] D. Wu, H. Zeng, C. Lu, and B. Boulet, "Two-stage energy management for office buildings with workplace ev charging and renewable energy," *IEEE Transactions on Transportation Electrification*, vol. 3, no. 1, pp. 225–237, March 2017.
- [156] Y. Guo, J. Xiong, S. Xu, and W. Su, "Two-stage economic operation of microgrid-like electric vehicle parking deck," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1703–1712, May 2016.
- [157] I. Momber, A. Siddiqui, T. G. S. Román, and L. Söder, "Risk averse scheduling by a pev aggregator under uncertainty," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 882–891, March 2015.
- [158] S. I. Vagropoulos and A. G. Bakirtzis, "Optimal bidding strategy for electric vehicle aggregators in electricity markets," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4031–4041, Nov 2013.
- [159] H. Wu, M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah, "A game theoretic approach to risk-based optimal bidding strategies for electric vehicle aggregators in electricity markets with variable wind energy resources," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 1, pp. 374–385, Jan 2016.
- [160] L. Yao, W. H. Lim, and T. S. Tsai, "A real-time charging scheme for demand response in electric vehicle parking station," *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 52–62, Jan 2017.
- [161] G. Binetti, A. Davoudi, D. Naso, B. Turchiano, and F. L. Lewis, "Scalable real-time electric vehicles charging with discrete charging rates," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2211–2220, Sept 2015.
- [162] Y. T. Liao and C. N. Lu, "Dispatch of ev charging station energy resources for sustainable mobility," *IEEE Transactions on Transportation Electrification*, vol. 1, no. 1, pp. 86–93, June 2015.
- [163] Q. Huang, Q. S. Jia, Z. Qiu, X. Guan, and G. Deconinck, "Matching ev charging load with uncertain wind: A simulation-based policy improvement approach," *IEEE Transactions on Smart Grid*, vol. 6, no. 3, pp. 1425–1433, May 2015.
- [164] R. Bellman, *Dynamic programming*. Princeton Univeristy Press, John Wiley & Sons, 1958.
- [165] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv:1506.00019*, 2015.

- [166] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.
- [167] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087–1088, 2018.
- [168] "California independent system operator." Accessed on May 5, 2017. [Online]. Available: <http://oasis.caiso.com/mrioasis/logon.do>
- [169] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 503–556, 2005.
- [170] Z. Wan, H. Li, and H. He, "Residential energy management with deep reinforcement learning," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–7.
- [171] H. Li, Z. Wan, and H. He, "Constrained EV charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, 2020.
- [172] 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.
- [173] F. Wei, Z. Wan, and H. He, "Cyber-attack recovery strategy for smart grid based on deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2476–2486, 2020.
- [174] F. Wei, Z. Wan, H. He, and X. Lin, "Ultrafast active response strategy against malfunction attack on fault current limiter," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2722–2733, 2020.
- [175] N. Xie, G. Ras, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *arXiv:2004.14545*, 2020.
- [176] "General data protection regulation (GDPR)." Accessed on June 20, 2020. [Online]. Available: <https://gdpr-info.eu>
- [177] K. Zetter. "Inside the cunning, unprecedented hack of ukraine's power grid." Accessed on June 20, 2020. [Online]. Available: <https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/>
- [178] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv:1412.6572*, 2014.
- [179] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv:1710.09282*, 2017.

BIBLIOGRAPHY

- “General data protection regulation (GDPR).” Accessed on June 20, 2020. [Online]. Available: <https://gdpr-info.eu>
- “California independent system operator.” Accessed on May 5, 2017. [Online]. Available: <http://oasis.caiso.com/mrioasis/logon.do>
- Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., *et al.*, “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *International Conference on Machine Learning*, 2016, pp. 173–182.
- Andreas, J., Rohrbach, M., Darrell, T., and Klein, D., “Neural module networks,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 39–48.
- Andreas, J., Rohrbach, M., Darrell, T., and Klein, D., “Learning to compose neural networks for question answering,” in *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2016, pp. 1545–1554.
- Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D., “VQA: Visual question answering,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2425–2433.
- Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K., “Purposive behavior acquisition for a real robot by vision-based reinforcement learning,” *Machine Learning*, vol. 23, no. 2, pp. 279–303, May 1996.
- Bahdanau, D., Cho, K., and Bengio, Y., “Neural machine translation by jointly learning to align and translate,” in *2015 International Conference on Learning Representations (ICLR)*, 2015.
- Baldi, P., “Autoencoders, unsupervised learning, and deep architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 37–49.
- Barth, A. L. and Poulet, J. F., “Experimental evidence for sparse firing in the neocortex,” *Trends in Neurosciences*, vol. 35, no. 6, pp. 345–355, 2012.
- Bellman, R., *Dynamic programming*. Princeton University Press, John Wiley & Sons, 1958.

- Bengio, Y., Courville, A., and Vincent, P., “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.
- Bengio, Y., *Learning deep architectures for AI*. Now Publishers Inc, 2009.
- Bengio, Y. and Delalleau, O., “On the expressive power of deep architectures,” in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 18–36.
- Beom, H. R. and Cho, H. S., “A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 3, pp. 464–477, Mar 1995.
- Binetti, G., Davoudi, A., Naso, D., Turchiano, B., and Lewis, F. L., “Scalable real-time electric vehicles charging with discrete charging rates,” *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2211–2220, Sept 2015.
- Bishop, C. M., *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.
- Boukas, E., Kostavelis, I., Gasteratos, A., and Sirakoulis, G. C., “Robot guided crowd evacuation,” *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 739–751, 2015.
- Brynjolfsson, E. and McElheran, K., “The rapid adoption of data-driven decision-making,” *American Economic Review*, vol. 106, no. 5, pp. 133–39, May 2016.
- Cao, G., Duan, Y., and Li, G., “Linking business analytics to decision making effectiveness: A path model analysis,” *IEEE Transactions on Engineering Management*, vol. 62, no. 3, pp. 384–395, 2015.
- Carriere, T. and Kariniotakis, G., “An integrated approach for value-oriented energy forecasting and data-driven decision-making application to renewable energy trading,” *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6933–6944, 2019.
- Chan, W., Jaitly, N., Le, Q., and Vinyals, O., “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4960–4964.
- Che, D., Safran, M., and Peng, Z., “From big data to big data mining: Challenges, issues, and opportunities.” Springer Berlin Heidelberg, 2013.
- Chen, C. P. and Zhang, C.-Y., “Data-intensive applications, challenges, techniques and technologies: A survey on big data,” *Information Sciences*, vol. 275, pp. 314 – 347, 2014.

- Chen, K., Hu, J., and He, J., “A framework for automatically extracting over-voltage features based on sparse autoencoder,” *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 594–604, 2018.
- Chen, K., Wang, J., Chen, L.-C., Gao, H., Xu, W., and Nevatia, R., “ABC-CNN: An attention based convolutional neural network for visual question answering,” *arXiv:1511.05960*, 2015.
- Chen, M., Mao, S., and Liu, Y., “Big data: A survey,” *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.
- Chen, Y., Li, W., Sakaridis, C., Dai, D., and Gool, L. V., “Domain adaptive Faster R-CNN for object detection in the wild,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 3339–3348.
- Chen, Y. F., Liu, M., Everett, M., and How, J. P., “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 285–292.
- Chen, Z., Jiang, C., and Guo, Y., “Pedestrian-robot interaction experiments in an exit corridor,” in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 29–34.
- Cheng, Y., Wang, D., Zhou, P., and Zhang, T., “A survey of model compression and acceleration for deep neural networks,” *arXiv:1710.09282*, 2017.
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B., “The cityscapes dataset for semantic urban scene understanding,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.
- Cybenko, G., “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.
- Dalal, N. and Triggs, B., “Histograms of oriented gradients for human detection,” in *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.
- Deb, K., “Multi-objective optimization,” in *Search Methodologies*. Springer, 2014, pp. 403–449.
- Deepmind. “Alphastar: Mastering the real-time strategy game starcraft II.” Accessed on September 22, 2020. [Online]. Available: <https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii>

- Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., “BERT: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- Duan, Y., Cui, B., and Yang, H., “Robot navigation based on fuzzy RL algorithm,” in *Advances in Neural Networks - ISNN 2008*, 2008, pp. 391–399.
- Eldridge, B. D. and Maciejewski, A. A., “Using genetic algorithms to optimize social robot behavior for improved pedestrian flow,” in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, 2005, pp. 524–529.
- Ernst, D., Geurts, P., and Wehenkel, L., “Tree-based batch mode reinforcement learning,” *Journal of Machine Learning Research*, vol. 6, no. Apr, pp. 503–556, 2005.
- Ferrer, G., Garrell, A., and Sanfeliu, A., “Robot companion: A social-force based approach with human awareness-navigation in crowded environments,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1688–1694.
- Field, D. J., “What is the goal of sensory coding?” *Neural Computation*, vol. 6, no. 4, pp. 559–601, 1994.
- Field, D., “Scale-invariance and self-similar ‘wavelet’ transforms: an analysis of natural scenes and mammalian visual systems,” in *Wavelets, Fractals and Fourier Transforms: New Developments and New Applications*, Oxford University Press, pp. 151–193, 1993.
- Frank, G. and Dorso, C., “Room evacuation in the presence of an obstacle,” *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 11, pp. 2135–2145, 2011.
- Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M., “Multimodal compact bilinear pooling for visual question answering and visual grounding,” in *Conference on Empirical Methods in Natural Language Processing*. ACL, 2016, pp. 457–468.
- Gaskett, C., Fletcher, L., and Zelinsky, A., “Reinforcement learning for a vision based mobile robot,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2000, pp. 403–409.
- Gong, M., Liu, J., Li, H., Cai, Q., and Su, L., “A multiobjective sparse feature learning model for deep neural networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3263–3277, Dec 2015.

- Goodfellow, I. J., Shlens, J., and Szegedy, C., “Explaining and harnessing adversarial examples,” *arXiv:1412.6572*, 2014.
- Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D., “Making the V in VQA matter: Elevating the role of image understanding in visual question answering,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6325–6334.
- Graves, A., Mohamed, A., and Hinton, G., “Speech recognition with deep recurrent neural networks,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J., “LSTM: A search space odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct 2017.
- Guo, Y., Xiong, J., Xu, S., and Su, W., “Two-stage economic operation of microgrid-like electric vehicle parking deck,” *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1703–1712, May 2016.
- Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L. A., *Feature extraction: foundations and applications*. Springer, 2008, vol. 207.
- He, K., Gkioxari, G., Dollár, P., and Girshick, R., “Mask R-CNN,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.
- He, K., Zhang, X., Ren, S., and Sun, J., “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, Sept 2015.
- He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.
- He, Y., Zhang, Z., Yu, F. R., Zhao, N., Yin, H., Leung, V. C. M., and Zhang, Y., “Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10 433–10 445, Nov 2017.
- He, Y., Zhao, N., and Yin, H., “Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, Jan 2018.
- Helbing, D., Buzna, L., Johansson, A., and Werner, T., “Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions,” *Transportation science*, vol. 39, no. 1, pp. 1–24, 2005.

- Helbing, D., Johansson, A., and Al-Abideen, H. Z., “Dynamics of crowd disasters: An empirical study,” *Physical review E*, vol. 75, no. 4, p. 046109, 2007.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R., “Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv:1207.0580*, 2012.
- Hochreiter, S. and Schmidhuber, J., “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- Huang, Q., Jia, Q. S., Qiu, Z., Guan, X., and Deconinck, G., “Matching ev charging load with uncertain wind: A simulation-based policy improvement approach,” *IEEE Transactions on Smart Grid*, vol. 6, no. 3, pp. 1425–1433, May 2015.
- Hubel, D. H. and Wiesel, T. N., “Receptive fields and functional architecture of monkey striate cortex,” *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.
- Hulme, O. J., Skov, M., Chadwick, M. J., Siebner, H. R., and Ramsøy, T. Z., “Sparse encoding of automatic visual association in hippocampal networks,” *NeuroImage*, vol. 102, pp. 458–464, 2014.
- Jan, B., Farman, H., Khan, M., Imran, M., Islam, I. U., Ahmad, A., Ali, S., and Jeon, G., “Deep learning in big data analytics: A comparative study,” *Computers & Electrical Engineering*, vol. 75, pp. 275–287, 2019.
- Jiang, C., Ni, Z., Guo, Y., and He, H., “Learning human-robot interaction for robot-assisted pedestrian flow optimization,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.
- Johansson, A., Helbing, D., Al-Abideen, H. Z., and Al-Bosta, S., “From crowd dynamics to crowd safety: a video-based analysis,” *Advances in Complex Systems*, vol. 11, no. 04, pp. 497–527, 2008.
- Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S. N., Rosaen, K., and Vasudevan, R., “Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1–8.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W., “Reinforcement learning: A survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
- Kaisler, S., Armour, F., Espinosa, J. A., and Money, W., “Big data: Issues and challenges moving forward,” in *2013 Hawaii International Conference on System Sciences*, 2013, pp. 995–1004.

- Kellerer, W., Kalmbach, P., Blenk, A., Basta, A., Reisslein, M., and Schmid, S., "Adaptable and data-driven softwarized networks: Review, opportunities, and challenges," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 711–731, 2019.
- Khan, N., Yaqoob, I., Hashem, I. A. T., Inayat, Z., Mahmoud Ali, W. K., Alam, M., Shiraz, M., and Gani, A., "Big data: survey, technologies, opportunities, and challenges," *The Scientific World Journal*, 2014.
- Kirkland, J. A. and Maciejewski, A. A., "A simulation of attempts to influence crowd dynamics," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, 2003, pp. 4328–4333.
- Kong, W., Dong, Z. Y., Hill, D. J., Luo, F., and Xu, Y., "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1087–1088, 2018.
- Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., and Zhang, Y., "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2019.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.
- Le, Q. V., "Building high-level features using large scale unsupervised learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 8595–8598.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.
- Lee, D., Conroy, M. L., McGreevy, B. P., and Barraclough, D. J., "Reinforcement learning and decision making in monkeys during a competitive game," *Cognitive Brain Research*, vol. 22, no. 1, pp. 45–58, 2004.
- Leistikow, D. "The egallon: How much cheaper is it to drive on electricity?" Accessed on August 16, 2017. [Online]. Available: <https://energy.gov/articles/egallon-how-much-cheaper-it-drive-electricity>
- Li, F., Niebles, J., and Savarese, S. "CS231n: Convolutional neural networks for visual recognition." Accessed on September 21, 2020. [Online]. Available: <https://cs231n.github.io/convolutional-networks/>
- Li, H., Wan, Z., and He, H., "Constrained EV charging scheduling based on safe deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, 2020.

- Li, H., Wan, Z., and He, H., “Real-time residential demand response,” *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 4144–4154, 2020.
- Liao, Y. T. and Lu, C. N., “Dispatch of ev charging station energy resources for sustainable mobility,” *IEEE Transactions on Transportation Electrification*, vol. 1, no. 1, pp. 86–93, June 2015.
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.
- Lipton, Z. C., Berkowitz, J., and Elkan, C., “A critical review of recurrent neural networks for sequence learning,” *arXiv:1506.00019*, 2015.
- Liu, H. and Motoda, H., *Feature extraction, construction and selection: A data mining perspective*. Springer Science & Business Media, 1998, vol. 453.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C., “SSD: Single shot multibox detector,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.
- Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E., “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.
- Long, M., Zhu, H., Wang, J., and Jordan, M. I., “Unsupervised domain adaptation with residual transfer networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 136–144.
- Lowe, D. G., “Object recognition from local scale-invariant features,” in *1999 IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 1999, pp. 1150–1157.
- Lu, J., Yan, Z., Han, J., and Zhang, G., “Data-driven decision-making (d3m): Framework, methodology, and directions,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 4, pp. 286–296, 2019.
- Lu, J., Yang, J., Batra, D., and Parikh, D., “Hierarchical question-image co-attention for visual question answering,” in *Advances In Neural Information Processing Systems (NIPS)*, 2016, pp. 289–297.
- Luh, P. B., Wilkie, C. T., Chang, S.-C., Marsh, K. L., and Olderman, N., “Modeling and optimization of building emergency evacuation considering blocking effects on crowd movement,” *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 687–700, 2012.

- Maaten, L. v. d. and Hinton, G., “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- Maigha and Crow, M. L., “Cost-constrained dynamic optimal electric vehicle charging,” *IEEE Transactions on Sustainable Energy*, vol. 8, no. 2, pp. 716–724, April 2017.
- Malinowski, M., Rohrbach, M., and Fritz, M., “Ask your neurons: A neural-based approach to answering questions about images,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1–9.
- Malinowski, M., Rohrbach, M., and Fritz, M., “Ask your neurons: A deep learning approach to visual question answering,” *International Journal of Computer Vision*, vol. 125, no. 1-3, pp. 110–135, 2017.
- Martinez-Gil, F., Lozano, M., and Fernández, F., “Marl-ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups,” *Simulation Modelling Practice and Theory*, vol. 47, pp. 259 – 275, 2014.
- Mhaskar, H., Liao, Q., and Poggio, T. A., “When and why are deep networks better than shallow ones?” in *Association for the Advancement of Artificial Intelligence*, 2017, pp. 2343–2349.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- Momber, I., Siddiqui, A., Román, T. G. S., and Söder, L., “Risk averse scheduling by a pev aggregator under uncertainty,” *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 882–891, March 2015.
- Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M., “Deepstack: Expert-level artificial intelligence in heads-up no-limit poker,” *Science*, vol. 356, no. 6337, pp. 508–513, 2017.
- Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E., “Deep learning applications and challenges in big data analytics,” *Journal of Big Data*, vol. 2, no. 1, 2015.
- Nam, H., Ha, J., and Kim, J., “Dual attention networks for multimodal reasoning and matching,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2156–2164.
- Namerikawa, T., Okubo, N., Sato, R., Okawa, Y., and Ono, M., “Real-time pricing mechanism for electricity market with built-in incentive for participation,” *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 2714–2724, Nov 2015.

- Neftci, E. O. and Averbeck, B. B., “Reinforcement learning in artificial and biological systems,” *Nature Machine Intelligence*, vol. 1, no. 3, pp. 133–143, 2019.
- Nixon, M. and Aguado, A., *Feature extraction and image processing for computer vision*. Academic press, 2019.
- Noh, H., Seo, P. H., and Han, B., “Image question answering using convolutional neural network with dynamic parameter prediction,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 30–38.
- Noh, H. and Han, B., “Training recurrent answering units with joint loss minimization for vqa,” *arXiv:1606.03647*, 2016.
- Olshausen, B. A. and Field, D. J., “Sparse coding with an overcomplete basis set: A strategy employed by v1?” *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.
- Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q., “Domain adaptation via transfer component analysis,” *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, Feb 2011.
- Pillai, J. R. and Bak-Jensen, B., “Integration of vehicle-to-grid in the western danish power system,” *IEEE Transactions on Sustainable Energy*, vol. 2, no. 1, pp. 12–19, Jan 2011.
- Pinheiro, P. O., “Unsupervised domain adaptation with similarity learning,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8004–8013.
- Poria, S., Cambria, E., Hussain, A., and Huang, G.-B., “Towards an intelligent framework for multimodal affective data analysis,” *Neural Networks*, vol. 63, pp. 104–116, 2015.
- Poultney, C., Chopra, S., Cun, Y. L., *et al.*, “Efficient learning of sparse representations with an energy-based model,” in *Advances in Neural Information Processing Systems (NIPS)*, 2006, pp. 1137–1144.
- Ren, M., Kiros, R., and Zemel, R., “Exploring models and data for image question answering,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2953–2961.
- Ren, S., He, K., Girshick, R., and Sun, J., “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

- Sakaridis, C., Dai, D., and Van Gool, L., “Semantic foggy scene understanding with synthetic data,” *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, Sep 2018.
- Schmidhuber, J., “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.
- Shu, M. and Fyshe, A., “Sparse autoencoders for word decoding from magnetoencephalography,” in *Proceedings of the third NIPS Workshop on Machine Learning and Interpretation in NeuroImaging*, 2013, pp. 1–7.
- Shu, Y., Cao, Z., Long, M., and Wang, J., “Transferable curriculum for weakly-supervised domain adaptation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4951–4958.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., *et al.*, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- Simonyan, K. and Zisserman, A., “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- Sun, K., Huang, S., Wong, D. S., and Jang, S., “Design and application of a variable selection method for multilayer perceptron neural network with LASSO,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1386–1396, 2017.
- Sutton, R. S. and Barto, A. G., *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.
- Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
- Tai, L., Paolo, G., and Liu, M., “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2017, pp. 31–36.
- Tajima, Y. and Nagatani, T., “Clogging transition of pedestrian flow in t-shaped channel,” *Physica A: Statistical Mechanics and its Applications*, vol. 303, no. 1, pp. 239–250, 2002.
- Tang, B., Jiang, C., He, H., and Guo, Y., “Human mobility modeling for robot-assisted evacuation in complex indoor environments,” *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 5, pp. 694–707, 2016.

- Tao, C., Pan, H., Li, Y., and Zou, Z., “Unsupervised spectral spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, Dec 2015.
- Toolan, T. M. and Tufts, D. W., “Detection and estimation in non-stationary environments,” in *Proceedings IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.
- Torrey, L., “Crowd simulation via multi-agent reinforcement learning,” in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010, pp. 89–94.
- Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T., “Adversarial discriminative domain adaptation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2962–2971.
- Vagropoulos, S. I. and Bakirtzis, A. G., “Optimal bidding strategy for electric vehicle aggregators in electricity markets,” *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4031–4041, Nov 2013.
- Van Hasselt, H., Guez, A., and Silver, D., “Deep reinforcement learning with double Q-learning,” in *Association for the Advancement of Artificial Intelligence*, 2016, pp. 2094–2100.
- Van Otterlo, M. and Wiering, M., “Reinforcement learning and markov decision processes,” in *Reinforcement Learning*. Springer, 2012, pp. 3–42.
- Vasquez, D., Okal, B., and Arras, K. O., “Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 1341–1346.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., “Attention is all you need,” in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.
- Wall, M. “Big data: Are you ready for blast-off?” Accessed on June 2, 2020. [Online]. Available: <https://www.bbc.com/news/business-26383058>
- Wan, Z. and He, H., “Weakly supervised object localization with deep convolutional neural network based on spatial pyramid saliency map,” in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 4177–4181.
- Wan, Z. and He, H., “Answernet: Learning to answer questions,” *IEEE Transactions on Big Data*, vol. 5, no. 4, pp. 540–549, 2019.

- Wan, Z., He, H., and Tang, B., “A generative model for sparse hyperparameter determination,” *IEEE Transactions on Big Data*, vol. 4, no. 1, pp. 2–10, 2018.
- Wan, Z., Hu, X., He, H., and Guo, Y., “A learning based approach for social force model parameter estimation,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 4058–4064.
- Wan, Z., Jiang, C., Fahad, M., Ni, Z., Guo, Y., and He, H., “Robot-assisted pedestrian regulation based on deep reinforcement learning,” *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1669–1682, 2020.
- Wan, Z., Li, H., and He, H., “Residential energy management with deep reinforcement learning,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–7.
- Wan, Z., Li, H., He, H., and Prokhorov, D., “Model-free real-time EV charging scheduling based on deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5246–5257, 2019.
- Wan, Z., Li, L., Li, H., He, H., and Ni, Z., “One-shot unsupervised domain adaptation for object detection,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.
- Wandel, M. “Bigger, tighter mouse maze experiments.” Accessed on January 12, 2020. [Online]. Available: https://woodgears.ca/farm/mouse_maze.html
- Wang, H., Xu, Z., Fujita, H., and Liu, S., “Towards felicitous decision making: An overview on challenges and trends of big data,” *Information Sciences*, vol. 367–368, pp. 747 – 765, 2016.
- Wang, L., “Heterogeneous data and big data analytics,” *Automatic Control and Information Sciences*, vol. 3, no. 1, pp. 8–15, 2017.
- Wei, F., Wan, Z., and He, H., “Cyber-attack recovery strategy for smart grid based on deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2476–2486, 2020.
- Wei, F., Wan, Z., He, H., and Lin, X., “Ultrafast active response strategy against malfunction attack on fault current limiter,” *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2722–2733, 2020.
- Wen, J., Zheng, N., Yuan, J., Gong, Z., and Chen, C., “Bayesian uncertainty matching for unsupervised domain adaptation,” *arXiv:1906.09693*, 2019.
- Wu, D., Zeng, H., Lu, C., and Boulet, B., “Two-stage energy management for office buildings with workplace ev charging and renewable energy,” *IEEE Transactions on Transportation Electrification*, vol. 3, no. 1, pp. 225–237, March 2017.

- Wu, H., Shahidehpour, M., Alabdulwahab, A., and Abusorrah, A., “A game theoretic approach to risk-based optimal bidding strategies for electric vehicle aggregators in electricity markets with variable wind energy resources,” *IEEE Transactions on Sustainable Energy*, vol. 7, no. 1, pp. 374–385, Jan 2016.
- Wu, L., Wang, Y., Li, X., and Gao, J., “Deep attention-based spatially recursive networks for fine-grained visual recognition,” *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1791–1802, 2019.
- Wu, Q., Wang, P., Shen, C., Dick, A., and v. d. Hengel, A., “Ask me anything: Free-form visual question answering based on knowledge from external sources,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4622–4630.
- Wu, Y., Chang, E. Y., Chang, K. C.-C., and Smith, J. R., “Optimal multimodal fusion for multimedia data analysis,” in *Proceedings of the 12th Annual ACM International Conference on Multimedia*, 2004, pp. 572–579.
- Xie, N., Ras, G., van Gerven, M., and Doran, D., “Explainable deep learning: A field guide for the uninitiated,” *arXiv:2004.14545*, 2020.
- Xu, H. and Saenko, K., “Ask, attend and answer: Exploring question-guided spatial attention for visual question answering,” in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 451–466.
- Xu, J., Xiang, L., Liu, Q., Gilmore, H., Wu, J., Tang, J., and Madabhushi, A., “Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 1, pp. 119–130, Jan 2016.
- Yamamoto, K. and Okada, M., “Control of swarm behavior in crossing pedestrians based on temporal/spatial frequencies,” *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 1036–1048, 2013.
- Yang, Z., He, X., Gao, J., Deng, L., and Smola, A., “Stacked attention networks for image question answering,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 21–29.
- Yao, L., Lim, W. H., and Tsai, T. S., “A real-time charging scheme for demand response in electric vehicle parking station,” *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 52–62, Jan 2017.
- You, K., Long, M., Cao, Z., Wang, J., and Jordan, M. I., “Universal domain adaptation,” in *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2720–2729.

- Zeiler, M. D. and Fergus, R., “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 818–833.
- Zetter, K. “Inside the cunning, unprecedented hack of ukraine’s power grid.” Accessed on June 20, 2020. [Online]. Available: <https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/>
- Zhang, J., Ding, Z., Li, W., and Ogunbona, P., “Importance weighted adversarial nets for partial domain adaptation,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8156–8164.
- Zhang, J., Springenberg, J. T., Boedecker, J., and Burgard, W., “Deep reinforcement learning with successor features for navigation across similar environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2017, pp. 2371–2378.
- Zhang, J., Klingsch, W., Schadschneider, A., and Seyfried, A., “Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2011, no. 06, p. P06004, 2011.
- Zhang, Q. and Zhao, D., “Data-based reinforcement learning for nonzero-sum games with unknown drift dynamics,” *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 2874–2885, 2019.
- Zhang, Q., Yang, L. T., Chen, Z., and Li, P., “A survey on deep learning for big data,” *Information Fusion*, vol. 42, pp. 146–157, 2018.
- Zhang, X., Zhao, J., and LeCun, Y., “Character-level convolutional networks for text classification,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 649–657.
- Zhang, Z., Zhao, D., Gao, J., Wang, D., and Dai, Y., “FMRQ-a multiagent reinforcement learning algorithm for fully cooperative tasks,” *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1367–1379, June 2017.
- Zhao, Y., Li, M., Lu, X., Tian, L., Yu, Z., Huang, K., Wang, Y., and Li, T., “Optimal layout design of obstacles for panic evacuation using differential evolution,” *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 175–194, 2017.
- Zhou, B., Tian, Y., Sukhbaatar, S., Szlam, A., and Fergus, R., “Simple baseline for visual question answering,” *arXiv:1512.02167*, 2015.
- Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A., “Target-driven visual navigation in indoor scenes using deep reinforcement

learning,” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3357–3364.