THE UNIVERSITY of ADELAIDE

Multi-modality Data Analysis Using Deep Reinforcement Learning

Hu Wang

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

March 26, 2025

Contents

List of Figures

List of Tables

University of Adelaide

Abstract

Multi-modality Data Analysis Using Deep Reinforcement Learning

by Hu Wang

Deep Reinforcement Learning (DRL) is a set of algorithms to deal with sequential decision problems. It mimics human learning behaviours through trial-and-error. Deep Reinforcement Learning has been developing steadily after decades of research and development. It has a wide range of applications that can deal with multi-modality data, such as robot navigation, traffic signal control, self-driving vehicles and etc. Recently, many impressive results have been achieved. However, it is not trivial to design a good state space, action space and reward function for reinforcement learning models to tackle a specific task. In this thesis, we propose a series of novel techniques to cope with challenging multi-modality data based on the idea of deep reinforcement learning.

We first proposed a DRL solution to alleviate the traffic congestion problem in urban areas. To achieve this, we introduce a DRL model with an edge-weighted graph convolutional encoder and unified structure decoder with states, actions and reward functions settings. Additionally, we also propose a new dataset and settings, including synthetic and real traffic data in more complex scenarios.

Behaviour cloning techniques directly copy the demonstrations from experts and it will inevitably incur worse performance on unseen environments due to error accumulation. Reinforcement learning based methods have better generalisation ability, but designing suitable reward functions towards a specific task is expensive. We thus propose a generic soft expert reward learning strategy to distil the expert's behaviour directly and successfully apply it on Vision-and-Language Navigation. The experimental results show the effectiveness of the model.

Finally, inspired by the exploration and expert's behaviour distillation of deep reinforcement learning, we propose a generic random network prediction strategy on anomaly detection and clustering tasks in an unsupervised learning manner. We propose to learn important features without using any labelled data by training neural networks to predict data distances in a randomly projected space.

Declaration of Authorship

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Hu Wang

May 2021

Acknowledgements

First and foremost, I would like to thank my Ph.D. supervisor, Professor Chunhua Shen. I would not finish my Ph.D. thesis successfully without his guidance and motivation. His brilliant ideas, foreseeing of research, professionalism and diligence towards working always inspire me in the research.

I want to thank Dr. Qi Wu. He is my co-supervisor in my Ph.D. career. His patient supports encourage me a lot to keep going on the road of research.

I would like to thank my closely worked collaborators. Guansong Pang, Jason Xue, Zhi Tian, Peng Chen and Massimiliano Zanin, thank them for every discussion. These inspired discussions really shine the light on my research. Dr. Wei Emma Zhang, it is my pleasure to work together with her on the course of Artificial Intelligence Technology.

My thanks go to all my supportive friends: Jianpeng Zhang, Yutong Xie, Yunzhi Zhuge, Libo Sun, Chunlei Liu, Zhi Tian, Wei Liu and Jiewei Cao. We had such a wonderful time here in Adelaide and I enjoy every moment spent with them. I would like to thank Jane Garrard and Rick Garrard for their thoughtful care that let me feel the warmth as with my families.

I own a deep sense of gratitude to my wife for her always companion and love without asking for a return. I am really a lucky man to be with her. Without her support, I could not go this far on the road of research. I sincerely thank my parents for their positive influence on me. Additionally, I felicitate myself on having such great support from my father-in-law and mother-in-law.

Last but not least, I would like to thank every friend in the Australian Institute for Machine Learning and the University of Adelaide for their support and help. They are the most brilliant fellows that I met and it is a great experience to work with them.

Chapter 1

Introduction

1.1 Reinforcement Learning Basis

In machine learning, Reinforcement Learning (RL) is a collection of algorithms. RL algorithms are designed to learn the strategy to take under certain circumstances (that is, how to map the current states into reasonable actions). Different from common supervised learning techniques, in reinforcement learning, the learner will not be told what action should be taken, but will try to find out which actions will generate the most profits. Put it formally, the goal of reinforcement learning is to learn a policy that makes decisions by the AI agents for each state to gain as much accumulated expected rewards as possible. For example, in a self-driving vehicle, the current observation of the AI driver is a state, and the agent tries to take the best action (turn left/right or brake) for the current observation; or the current chessboard is a state, and we would like to know which location to put the next piece is the best choice.

In machine learning, supervised learning is to learn knowledge from the labelled training set provided by external supervisors. Unsupervised learning is to learn an implicit structure of the unlabelled data. Reinforcement learning is the third machine learning paradigm that parallels the above two. Concretely, RL agents utilise past experiences to assist current situations to select the most suitable actions to gain rewards through trial-and-error. But it has connections with supervised learning and unsupervised learning as well. For discrete actions, the probability of actions is a multinomial distribution. If we regard the action as a label to the corresponding state and treat different state-action pairs as independent and identically distributed (i.i.d)., supervised learning (i.e., imitation learning) can be adopted to deal with such sequential decision problems. In this case, the learned policy is similar to a classifier or a regressor towards different states.

RL tasks usually can be modelled as Markov Decision Processes (MDPs). When an agent is in an environment, after an action is performed by the agent, the environment will transfer to another state accordingly with certain probabilities. Later on, rewards will be returned according to the action. The agent is able to update the model with the given rewards for better action choosing. Therefore, a close-loop of RL model

optimisation is formed. In summary, reinforcement learning mainly includes four elements: states, actions, transition probabilities, and reward functions.

1.1.1 Markov Process

Markov process (MP) is a process that obeys the Markov property, in which the conditional probability of the next state is only relying on the current state of the system, and is independent and irrelevant to its history or future state Britannica, 2006. The stability of the Markov process ensures that the future can be predicted through past experiences. A Markov process with discrete states is usually called a Markov chain. At each step of the Markov process, the system can change from one state to another according to the probability distribution, or maintain the current state. The change of state is called transition, and the probabilities concerning different state changes are called transition probabilities. Markov chains are usually defined by discrete-time sets, also known as discrete-time Markov chains Billingsley, 1961; but continuous time is adopted as well, termed as continuous-time Markov chain.

Formally, the Markov property can be presented as a two-tuple $\langle \mathcal{S}, \mathcal{P} \rangle$:

$$\begin{aligned} &\mathbb{P}\left\{\mathcal{S}\left(t_n\right)=S_n \mid \mathcal{S}\left(t_1\right)=S_1, \mathcal{S}\left(t_2\right)=S_2, \cdots, \mathcal{S}\left(t_{n-1}\right)=S_{n-1}\right\} \\ =&\mathbb{P}\left\{\mathcal{S}\left(t_n\right)=S_n \mid \mathcal{S}\left(t_{n-1}\right)=S_{n-1}\right\}, \end{aligned} \tag{1.1}$$

where $t_1, t_2, t_3, ..., t_n$ is a time series and $\mathcal{S}(t)$ is a random variable. $\mathcal{P}$ is the transition probability.

From this definition, we can see that the future time t_n is only affected by the current time t_{n-1}, and has nothing to do with the previous t_1 to t_{n-2}. In other words, the current state captures all sufficient information to determine the next state, and all previous states can be cast away. Thus, Markov property is also known as non-aftereffect or memorylessness property Shannon, 1948. Brownian motion, Random walk and the transferring process of thermal electron are typical examples of continuous-time Markov processes Bharucha-Reid, 1997.

1.1.2 Markov Reward Process

On the basis of the Markov process, a reward value is added next to each state, indicating the expected rewards an agent can get from reaching that state to the end. It constitutes a Markov reward process (MRP). In the Markov reward process, the added rewards are presented as $\mathcal{R}$ and a decay discount factor γ is further attached to the future reward over time. Formally, the Markov reward process can be represented as a four-tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$.

Inside the four-tuple, $\mathcal{S}$ represents the state set; $\mathcal{P}$ represents the state transition probability matrix $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$; $\mathcal{R}$ is the reward function, defined as $\mathcal{R}_s = \mathbb{E}[R_{t+1} | \mathcal{S}_t = s]$ means the expected reward that can be obtained at the next

moment $t+1$ under the current state t of $\mathcal{S}_t$; γ is the discount factor, and γ is under the constrain $\gamma \in [0,1]$. The closer the γ to 0, the more preference of the model towards the instant rewards; vice versa. The existence of reward discount is because the later steps may contain more uncertainty to gain the rewards, as well as it can successfully deal with reward dead-looping issue. The discounted accumulated return G_t from time step t can be formally presented as:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \end{aligned} \tag{1.2}$$

In the Eq. 1.2, the discounted accumulated reward G_t represents the reward accumulated from a single trajectory. Therefore, a direct summation of received rewards from each step is performed without any expectation operation.

Value Function in Markov reward process

Value Function Silver, 2015 in the Markov reward process is defined as the expected accumulated reward that can be gained from one state. Intuitively, it indicates the "potential" (may not gain exactly that number of rewards at the end of each episode, but averagely and potentially the agent could gain) returns when the agent is at that state. In order to depict the expected reward that can be gained, the expectation operation over all random trajectories is performed. Formally, the Value Function can be presented as:

$$\begin{aligned} v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \dots\right) \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma v\left(S_{t+1}\right) \mid S_t = s\right], \end{aligned} \tag{1.3}$$

One thing worth to be noted here: the value function in the Markov reward process is a bit different from the value function in the Markov decision process. In the Markov reward process, there have no actions, there thus is no concept of policy. Consequently, the agent would just follow the transition probabilities to traverse to another state (equivalent to a random policy that has the same probabilities to follow each action).

Bellman Equation

The last step of the value function is the basic form of the Bellman equation. From the formula, the value of the current state is related to the current feedback and the value of the next step. It shows that Value Function can be calculated iteratively. The last step of the value function can be further denoted as:

$$v(s) = \sum_{s',r} \mathbb{P}(s', r \mid s)[R_s + \gamma v\left(s'\right)], \tag{1.4}$$

where the probability $\mathbb{P}(s', r \mid s)$ is the transition probability from state s to state s' with reward. Note here that the fetch of R_s is with certain probabilities, thus we would calculate the marginal probabilities by averaging over all s' and r (we discuss finite states and actions here).

1.1.3 Markov Decision Process

Reinforcement learning tasks usually can be described as a Markov Decision Process (MDP), which formally describes an environment combined with actions available. To put it simply, MDP is a cyclical process in which an agent takes actions to change its states to obtain rewards from the environment. The strategy of the MDP depends entirely on the current state (Only present matters), which is also a manifestation of its Markov property. Formally, the Markov decision process can be represented as a five-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. Compared with the Markov reward process, the action space $\mathcal{A}$ is added into the Markov decision process.

More concretely, when an agent is spawn in an environment, we assume the agent can fully observe the state space $\mathcal{S}$, where each state $s \in \mathcal{S}$. The actions that the agent can take constitute the action space $\mathcal{A}$; if a certain action $a \in \mathcal{A}$ acted by the agent on the current state s, the state will be transferred to one of the next states s' with the action depended transition probability matrix $\mathcal{P}^a_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$; at the mean time, a reward $r \in \mathcal{R}$ is returned and the reward is action-dependent as well $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$.

State-value Function

Different from the state value function defined in the Markov reward process, the state-value function in MDP takes the policy π into consideration. It defined as the expected accumulated reward starting from state s and follows the policy π.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s\right]. \end{aligned} \tag{1.5}$$

Bellman Expectation Equation for state-value is defined as the average over all possible q values following policy π:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) q_\pi(s, a). \tag{1.6}$$

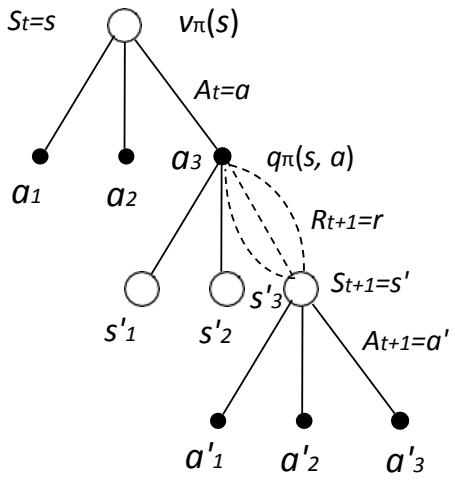

FIGURE 1.1. An illustration of the process of the Markov decision process. It also shows the reason why the Bellman expectation equation is calculated as shown in Eq. 1.6 and Eq. 1.8. Note that the dashed lines of rewards represent the receiving of rewards is based on probabilities. For the Bellman expectation equation of state-value, it is the average over all actions; while for action-value, the Bellman expectation equation is the average over the MDP dynamic probabilities given current state and action.

Action-value Function

Similar to the state-value function, the action-value function considers the policy as well. It defined as the expected accumulated reward starting from state s, then taking action a and follows the policy π.

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\ &= \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi \left(S_{t+1}, A_{t+1} \right) \mid S_t = s, A_t = a \right]. \end{aligned} \tag{1.7}$$

Bellman Expectation Equation for action-value Similar as the Bellman Equation in MRP. It is defined as the average over all transition probabilities covered by the environment:

$$q_\pi(s,a) = \sum_{s',r} \mathbb{P}\left(s', r \mid s, a\right) \left[r + \gamma v_\pi \left(s'\right) \right]. \tag{1.8}$$

The above shown two Eq. 1.6 and Eq. 1.8 are the Bellman expectation equations for state-value and action-value. But it contains each other. In order to make them in better representations, we can make a simple substitution. For state-value function:

$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s) \sum_{s',r} \mathbb{P}\left(s', r \mid s, a\right) \left[r + \gamma v_\pi \left(s'\right) \right]. \tag{1.9}$$

For action-value function, we have:

$$q_\pi(s,a) = \sum_{s',r} \mathbb{P}\left(s', r \mid s, a\right) \left[r + \gamma \sum_{a'} \pi\left(a' \mid s'\right) q_x \left(s', a'\right) \right]. \tag{1.10}$$

In this case, the above two equations only contains v and q, respectively.

The connections with MP and MRP

The MDP is strong connected with Markov process and Markov reward process as stated by Silver, 2015. Interestingly, the state sequence $S_1, S_2, S_3, ...$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$. Similarly, the state and reward sequence $S_1, R_1, S_2, R_2, S_3, R_3, ...$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$. But worth to be noted here, the transition probability matrix and the reward function depend on the policy π (different from random policy case stated in the above MRP section). the transition probability matrix and the reward function of MRP are the average of corresponding value of MDP over the policy probabilities. More formally, we have:

$$\begin{aligned} \mathcal{P}_{ss'}^{\pi} &= \sum_{a \in \mathcal{A}} \pi(a \mid s) \mathcal{P}_{ss'}^{a} \\ \mathcal{R}_{s}^{\pi} &= \sum_{a \in \mathcal{A}} \pi(a \mid s) \mathcal{R}_{s}^{a}. \end{aligned} \tag{1.11}$$

Bellman Optimality Equation

The optimal state-value function is defined as the max value across all actions given current state:

$$\begin{aligned} v_*(s) &= \max_{\pi} v_{\pi}(s) \\ &= \max_{a} q_*(s, a). \end{aligned} \tag{1.12}$$

Similarly, the optimal action-value function is defined as the max q-value across all actions given current state:

$$\begin{aligned} q_*(s, a) &= \max_{\pi} q_{\pi}(s, a) \\ &= \sum_{s:r} p\left(s', r \mid s, a\right) \left[r + \gamma v_*\left(s'\right)\right]. \end{aligned} \tag{1.13}$$

Let the optimal policy π_*

$$\pi_* = \arg\max_{\pi} v_{\pi}(s) = \arg\max_{\pi} q_{\pi}(s, a). \tag{1.14}$$

Thus, after simple substitution, we have the equations for optimal policy π_*

$$V_*(s) = \max_{a} \sum_{s', r} p\left(s', r \mid s, a\right) \left[r + \gamma v_*(s)\right], \tag{1.15}$$

and

$$q_*(s, a) = \sum_{s', r} p\left(s', r \mid s, a\right) \left[r + \gamma \max_{a'} q_*\left(s', a'\right)\right]. \tag{1.16}$$

The two equations above are Bellman Optimality Equations.

1.1.4 Multi-armed Bandit

The Multi-armed Bandit (MAB) problem is a special case of MDP with only one state and a single-step decision process (the choice of action will not affect the state shown in the next round). The problem setting Vermorel and Mohri, 2005 is there are multiple pokie machines in a gaming room. A gambler chooses an arm and pulls it at one time and the arm has a certain probability of spitting out bonus coins. However,

the gambler does not know how this probability comes. The goal is to find a strategy for the gambler to maximise the bonus by pulling different arms. Similar to multi-step reinforcement learning scenarios, if we would like to maximise a single-step sequential decision process, we ought to know the expected reward of each action, and then choose the action with the largest expected reward.

If we hold the assumption that the reward for each action is fixed, we are able to fetch the rewards of actions by pulling them only once. However, this is too ideal for bandit settings since the obtaining rewards of actions obey certain probabilities in MAB. In this case, multiple attempts are required. Intuitively, similar to all the reinforcement learning algorithms, the basic idea of the MAB algorithm is through trial-and-error. If a certain choice yields a better result, then we should choose it more, and vice versa. A naive bandit algorithm is relatively simple. The practical steps are: after each arm has been selected to calculate a mean, always choose the one with the largest mean.

The naive bandit algorithm tries to maximise expected reward, but it introduces bias by neglecting other potential profitable arms. In this case, an arm may not be selected for the entire game if it does not perform well at the first several rounds. If a certain arm has been chosen too few times, leading to uncertainties, we thus should give it more chances to be selected until it is determined to be gold or stone. In a word, more opportunities should be given to those arms that are profitable or uncertain. ϵ-greedy is a typical solution for the exploitation vs. exploration problem Vermorel and Mohri, 2005. More specifically, a small positive number between (0, 1) will be given as the initial ϵ. Each time to choose an agent, a number is randomly generated. If it is smaller than epsilon, the agent will select an arm randomly; otherwise, the arm with the largest expected bonus will be selected. The closer epsilon to 0, the more conservative to choose a random arm; vice versa.

Besides the ϵ-greedy algorithm, Upper Confidence Bound (UCB) Auer, 2002 is another algorithm for the bandit problem. It enables the arm to choose based on a pre-defined equation. Intuitively, the larger the mean but the smaller the standard deviation, and the greater the probability of this arm being selected, which serves as exploitation; meanwhile, arms that have been selected few times can also get chances to be selected, which plays the role of exploration.

Differently, Thompson sampling Agrawal and Goyal, 2012 follows a Bayesian idea to deal with the bandit problem. It assumes the probability of an arm spitting out bonus coins obeys a prior distribution. For example, a very common one is the Beta distribution. After actually choosing different arms, we can thus adjust the corresponding posterior distributions based on the corresponding feedbacks. Later on, the arms can be chosen based on the fitted posterior probabilities.

1.2 Deep Learning Basis

1.2.1 Convolutional Neural Networks

In recent years, deep learning techniques have gained tremendous momentum with the rising of computation power. Convolutional Neural Network (CNN) is currently one of the powerful deep learning neural network structures, and has achieved stunning performance in computer vision and image recognition applications. Early in the 80s and 90s of the 20th century. The prototype of CNNs was proposed Fukushima and Miyake, 1982; Denker et al., 1989. Later on, back-propagation was introduced into CNN models by LeCun LeCun et al., 1989; LeCun et al., 1998, which greatly speed up the process. It was mainly used for text recognition at that time. After enormous efforts have been spent to optimize the CNN structure and thanks to the surge of computational resources, as well as large-scale datasets Deng et al., 2009; Lin et al., 2014, the CNNs become the models that we widely use today Krizhevsky, Sutskever, and Hinton, 2012; He et al., 2016.

By analysing the characteristics of images, CNNs introduce a prior knowledge (sometimes it is called inductive bias) named convolutional kernel to consider neighbouring pixels. Compared with Multi-layer Perceptron (MLP) models Rumelhart, Hinton, and Williams, 1985, CNNs can greatly reduce model parameters by sharing the convolution kernels. The CNN models have translation invariance as well. In the scenario of reinforcement learning, CNN models usually work as feature extractors for obtaining essential representations from raw image states, e.g. raw images can be inputted as the states of RL agents in some Atari games Mnih et al., 2013.

1.2.2 Recurrent Neural Networks

Different from Convolutional Neural Networks, Recurrent Neural Networks (RNNs) Rumelhart, Hinton, and Williams, 1986 is another subset of Neural Networks that can deal with the sequential decision process in a good manner. Despite CNN models are able to process sequential data Zhang and Wallace, 2015 as well. A sequence structure will be obtained after expanding an RNN model. Within the model, the previous output will be served as the next input. In such a case, the previous outputs will influence the inputs of the next neuron. This chain feature reveals that RNN is essentially related to the sequence. Thus, usually RNNs are used for natural language text processing or human speech processing. But naive RNNs are facing the problem of retaining long-distance relations. In order to solve this issue, LSTM Hochreiter and Schmidhuber, 1997 that is a special form of RNN, is proposed by introducing different controlling gates.

Imitation Learning (IL) is another family of algorithms for the sequential decision process. Put it in the imitation learning scenarios, RNNs can also be used to learn policies Sun et al., 2017 due to their excellent power to deal with sequential data.

1.3 Motivation, Contributions and Thesis Outline

As mentioned above, Deep reinforcement learning leverages the latest advances in both deep learning and reinforcement learning. Thus, it has a great potential to provide solutions to a number of applications to deal with data of multi-modality. To excavate the ability of deep reinforcement learning for multi-modal data analysis, in this thesis, we design a number of simple yet efficient models for various task based on the idea of deep reinforcement learning, including traffic signal control, vision-and-language navigation and more general anomaly detection/clustering tasks. The contributions of this thesis are:

- We first proposed a deep reinforcement learning solution to alleviate the traffic congestion problem in urban areas. To achieve this, a DRL model with an edge-weighted graph convolutional encoder and unified structure decoder is adopted, as well as ideal states, actions and reward functions are designed. From our experiments, we empirically find that jammed traffic flows can be optimised effectively with this model. Moreover, along with the DRL model, we also propose a new dataset and settings, including both synthetic and real traffic data in more complex scenarios, since the traffic settings of existing methods are usually simplified and inconsistent.
- In sequential decision tasks, supervised learning based techniques directly copy the experts' behaviours and it will inevitably incur worse performance on the inference phase due to the distribution shift issue. Reinforcement learning based methods have better generalisation ability. Nevertheless, designing suitable reward functions for a specific task is non-trivial. In this case, we propose a generic soft expert reward learning strategy and successfully apply it to the Vision-and-Language Navigation task. With the soft expert reward learning module, agents can distil the experts' policy directly without explicitly specifying the reward function by comparing the state-behaviour pair of agents and experts in a latent space. Along with the soft expert reward learning module, a self perceiving module targeting pushing the agent towards the final destination as fast as possible is further proposed.
- Finally, inspired by the idea of exploration within deep reinforcement learning, we apply the random network prediction to anomaly detection and clustering tasks in an unsupervised learning manner. We propose to learn important features without using any labelled data by training neural networks to predict data distances in a randomly projected space. In order to well predict random distances, the representation learner is forced to learn the structures of data. The experimental results demonstrate the effectiveness of the proposed method.

The thesis is organised as follows:

In Chapter 1, we provide the problem definition and basis of reinforcement learning and deep learning.

In Chapter 2, detailed literature reviews on different deep reinforcement learning algorithms with the application are given.

In Chapter 3, a benchmark dataset and a novel edge-weighted graph convolutional encoder and unified structure decoder DRL model are introduced.

In Chapter 4, we propose a soft expert reward learning for vision-and-language navigation task to distil expert's policy directly from demonstrations of experts.

In Chapter 5, we further apply the random network prediction into anomaly detection and clustering tasks. A simple yet effective random distances prediction model is proposed to reserve data point structures.

In Chapter 6, we finally summarise the thesis and the potential research directions are discussed.

Chapter 2

Literature Review

2.1 Imitation Learning

Similar to the common supervised learning framework that treats training data obeying the independent and identically distributed assumption (i.i.d.), Imitation Learning (IL) is another family of algorithms to deal with sequential decision problems. IL models treat expert data as strong supervised signals and they directly learn observation-action pairs from expert's demonstrations.

IL techniques enable the agents to perform better on seen environments but may generalise poorly to unseen ones due to the error accumulation Reddy, Dragan, and Levine, 2019. The policy will shift away gradually during the inference time since the states in the testing may not be exactly the same as them in the training phase. Thus, some IL frameworks, such as DAgger Ross, Gordon, and Bagnell, 2011 and its variants, incorporate human efforts into the data collection process. By iteratively collecting data from the real trajectories of agents and labelling the collected data, it solves the error accumulation problem mathematically, but it significantly increases human labours.

We meet the distribution shift problem in the vision-and-language navigation task by applying simple behaviour cloning. RL models have better generalisation ability, but it is non-trivial to design a suitable reward function. In order to solve the issue, we design a soft expert reward learning to directly distil reward functions from expert's demonstrations.

2.2 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a combination of deep learning and reinforcement learning. It integrates the powerful fitting abilities of deep neural networks with the decision-making ability of reinforcement learning to achieve end-to-end learning (although it is different from gradient back-propagation of supervised learning to minimise the error of an objective function). The emergence of DRL makes reinforcement learning practical to solve complex problems in real-world scenarios. Some successful

algorithms include Policy Gradient Sutton et al., 1999, Deep-Q-learning Mnih et al., 2013, Actor-Critic Konda and Tsitsiklis, 2000, Deep Deterministic Policy Gradient (DDPG) Lillicrap et al., 2015 and Proximal policy optimisation (PPO) Schulman et al., 2017, etc.

2.2.1 Model-free Reinforcement Learning

Model-free Reinforcement Learning is to learn the optimal strategies without knowing the transition probabilities of the Markov decision process. More intuitively, the agent directly models the posterior probability of action given state without knowing about how s becomes s' of the current environment.

Policy-based Reinforcement Learning

Policy-based reinforcement learning directly models the strategy, that is, given the state, the selected action is directly obtained instead of being derived from the value function. Policy Gradient (PG) Sutton et al., 1999 is a typical policy-based reinforcement learning algorithm. PG is not based on back-propagate through errors as supervised learning algorithms do. Instead, it optimises the policy based on the feedbacks from the environment. Intuitively, if an action gets more rewards, then we increase the probability of its appearance; if an action gets fewer rewards, we decrease its probability of appearance. We can propagate the gradients into the policy networks towards the direction to increase the expectation of rewards by:

$$\begin{aligned}
\nabla_\Theta \mathbb{E}[r(s,a)] &= \nabla_\Theta \sum \pi(a \mid s,\Theta) r(s,a) \\
&= \sum \nabla_\Theta \pi(a \mid s,\Theta) r(s,a) \\
&= \sum \pi(a \mid s,\Theta) \frac{\nabla_\Theta \pi(a \mid s,\Theta)}{\pi(a \mid s,\Theta)} r(s,a) \\
&= \sum \pi(a \mid s,\Theta) \nabla_\Theta \log \pi(a \mid s,\Theta) r(s,a) \\
&= \mathbb{E}\left[\nabla_\Theta \log \pi(a \mid s,\Theta) r(s,a)\right]
\end{aligned} \tag{2.1}$$

Thus, the overall objective function to be minimised can be presented as:

$$L(\Theta) = -\sum \log \pi(a \mid s,\Theta) r(s,a), \tag{2.2}$$

where the Θ is the parameters to be optimised and $r(\cdot)$ is the reward/evaluation function. The above objective can be directly optimised by gradient decent. Note that the evaluation function do not have to be differentiable.

Value-based Reinforcement Learning

Bellman Optimality Equations build strong foundations for value-based reinforcement learning algorithms. Different from policy-based reinforcement learning techniques, value-based reinforcement learning models thus try to distil policies from value functions directly. The most widely used value-based RL method is DQN Mnih et al., 2013. In DQN, the agent chooses action purely based on the Bellman optimality equation for action-value. In order to stabilise the training process and increase the sample efficiency of DQN, target network and experience replay Mnih et al., 2015 techniques are proposed, respectively. Multiple DQN variants Van Hasselt, Guez, and Silver, 2016; Wang et al., 2016; Schaul et al., 2015 have been proposed as well. Specifically, Double DQN Van Hasselt, Guez, and Silver, 2016 avoids over-estimated the q-value by separating the action selection network and the evaluation network. Wang et al., 2016 proposed a duelling network to divide the model output into two branches: one branch predicts the state-values and the other network predicts advantages for each action, respectively. Then, the q-values can be formed by summing the two outputs to reduce the high variance of the value function estimation. Different from DQN or its off-policy variants, SARSA Rummery and Niranjan, 1994 is an on-policy RL algorithm. SARSA will actually take the selected action following the current ϵ-greedy strategy, and estimate the q-values based on the action and next state in an online manner. Worth to be noted here, compared to algorithms of the DQN family, the action selection of SARSA is more conservative. Prioritised experience replay (PER) Schaul et al., 2015 adopts a hard example mining strategy to give more priorities to those experiences with higher Temporal difference error (TD-error).

The Combination of Policy and Value

Actor-critic (AC) algorithm Konda and Tsitsiklis, 2000 is a typical RL technique to combine policy-based RL and value-based RL. As its name, Actor-critic means that an actor is acting; while at the same time a critic is giving advice. The two networks would promote each other and thus be optimised together (this idea shares some similarities with Generative Adversarial Networks Goodfellow et al., 2014). AC is an on-policy algorithm and it is able to deal with continuous action space that DQN feels difficult to tackle. Asynchronous Advantage Actor-Critic (A3C) Mnih et al., 2016 is an algorithm based on the AC framework, but with the idea of multi-thread to assign different agents to experience different trajectories simultaneously and update the central model with the gradients collected by worker agents. Deep Deterministic Policy Gradient (DDPG) Lillicrap et al., 2015 algorithm combines the AC framework and the idea of DQN based action value to deal with the continuous control problem. Similar to the target network in DQN, it introduces each target network for actor and critic, respectively. In DDPG, the policy can be updated in a TD manner through off-policy learning to predict a deterministic action. Trust region policy optimisation (TRPO) Schulman et al., 2015 and Proximal policy optimisation algorithms (PPO)

Schulman et al., 2017 are two RL algorithms based on the idea of approximation-maximisation and importance sampling. They constrain the new parameters or the new outputted policy distribution in a small trust-region compared to the old ones. The model can thus be updated in an off-policy manner. Rainbow shows that by iteratively adopting different RL algorithms can receive promising results.

2.2.2 Model-based Reinforcement Learning

Model-based RL algorithms aim to improve RL learning efficiency by building an internal environment transition model. In 1991, Sutton proposed Dyna Sutton, 1991. It is a generic framework that is applied to various existing model-free algorithms. Dyna combines with RL algorithms and follows the idea that attempts the policy in reality once and try it K times in a virtual environment. Model-predictive control (MPC) Agachi et al., 2016 is a classic optimal control technique based on planning to select actions. In MPC, the agent observes a state at each step and it will output an optimal action for the current model. Levine et al. proposed Guided policy search (GPS) Levine and Koltun, 2013 to iteratively fit the environment model and the policy. Imagination-augmented agents (I2A) Racanière et al., 2017 is another framework combining model-based RL and model-free RL. I2A mainly improves data efficiency and model performance/robustness with an imperfect environment model. by simulating the process of human thinking to consider several steps forward. World Models Ha and Schmidhuber, 2018 includes a VAE-based visual module to compresses raw images into low-dimensional representations; an RNN-based memory module to predict the future state based on historical information; a decision-making module to choose an action only based on the representations of the visual module and the memory module.

2.3 Exploration Strategies of Reinforcement Learning

As mentioned above, ϵ-greedy is a simple yet efficient exploration strategy and it has been adopted in many deterministic-policy RL algorithms Mnih et al., 2013; Van Hasselt, Guez, and Silver, 2016; Wang et al., 2016. For non-deterministic policy RL algorithms Konda and Tsitsiklis, 2000; Mnih et al., 2016, agents select actions based on certain probabilities, which allows exploration of different actions under different circumstances. However, they may meet difficulties Burda et al., 2019 while dealing with complex problems (e.g. Montezuma's Revenge game) that require long trajectories of actions. In order to solve this issue, some work Pathak et al., 2017; Burda et al., 2019 introduce intrinsic rewards to encourage agents to explore unknown areas and receive promising results in those complicated environments. Soft Actor-Critic (SAC) Haarnoja et al., 2018 motivate the exploration behaviour by adding an entropy term on the action domain. Inspired by the exploration strategies in RL, we further propose a generic RDP model by applying it to anomaly detection and clustering tasks in an unsupervised learning manner.

2.4 Applications of Reinforcement Learning

Some successful applications include Deep-Q-learning Mnih et al., 2013 to play Atari games; AlphaGo Silver et al., 2017 for Go game-playing; AlphaStar Vinyals et al., 2019 for multi-agent control within StarCraft; autonomous driving vehicle control with DRL Sallab et al., 2017, etc.

Nevertheless, the design of good states, actions and rewards are challenging for different tasks. Following the vein of RL applications, we propose to apply a reinforcement learning algorithm to traffic signal control and visual-and-language navigation tasks. Besides that, we further propose novel solutions towards the characteristics of the applied problems. Extensive experimental results demonstrate the effectiveness of the proposed models.

Chapter 3

Multi-intersection Traffic Optimisation: A Benchmark Dataset and a Strong Baseline

3.1 Overview

The control of traffic signals is fundamental and critical to alleviating traffic congestion in urban areas. However, it is challenging since traffic dynamics are complicated in real-world scenarios. Because of the high complexity of the optimisation problem for modelling traffic, the experimental settings of existing works are often inconsistent. Moreover, it is not trivial to control multiple intersections properly in real complex traffic scenarios due to its vast state and action space. Failing to take intersection topology relations into account also results in inferior solutions. To address these issues, in this chapter we carefully design our settings and propose a new dataset including both synthetic and real traffic data in more complex scenarios. Additionally, we propose a novel baseline model with strong performance. It is based on deep reinforcement learning with an encoder-decoder structure: an edge-weighted graph convolutional encoder to excavate multi-intersection relations; and a unified structure decoder to jointly model multiple junctions in a comprehensive manner, which significantly reduces the number of the model parameters. By doing so, the proposed model is able to effectively deal with the multi-intersection traffic optimisation problem. Models are trained/tested on both synthetic and real maps and traffic data with the Simulation of Urban Mobility (SUMO) simulator. Experimental results show that the proposed model surpasses multiple competitive methods.

3.2 Introduction

Nowadays, traffic congestion has become a practical but challenging issue in our daily lives. It not only costs a significant amount of driver's waiting time, but also causes air pollution. To address these issues, expanding road networks is a solution, but it is

not always feasible due to various constraints. Developing new intelligent algorithms for traffic control offers an alternative but more economical choice.

Traffic optimisation is an active research area, where many existing works have been proposed for efficient traffic signal control Li, Wen, and Yao, 2014; Baluja, Covell, and Sukthankar, 2017; Covell, Baluja, and Sukthankar, 2015; Liang et al., 2018. These works can be roughly classified into two categories: predefined rule-based adaptive methods; learning-based methods such as hill-climbing Baluja, Covell, and Sukthankar, 2017 or deep reinforcement learning (DRL) based methods which attempt to fit the dynamic fluctuation of traffic flows Liang et al., 2018. However, in the aforementioned works, the simulation settings and datasets are usually inconsistent. Additionally, many current works consider traffic dynamics of single or few junctions. These methods show promising results under the simplified situations, but it may not generalise well to real traffic dynamics.

In order to tackle these issues, here we carefully design our settings/data and propose a strong baseline—the Edge-weighted Graph convolutional and unified structural Reinforcement Learning (EGU-RL) model. Our proposed EGU-RL model is able to control large-scale traffic junctions with a unified central agent. It can further take advantage of relations between junctions with edge-weighted graph convolutional networks by taking the spatial and distance information into account. The main contributions of this work are as follows.

- We carefully design our settings and propose a new dataset containing one synthetic map and multiple real maps from the city of Manhattan and Suzhou. Along with these maps, both synthetic and real traffic flows in comprehensive and complex scenarios are provided.
- We introduce a grouped multi-agent reinforcement learning baseline model to group neighbouring intersections, which is able to alleviate the drawbacks of exponentially increased action dimensions.
- We propose a novel and strong baseline model. We introduce a reinforcement-learning model with an edge-weighted graph convolutional encoder and a unified structure decoder to optimise multi-intersection flows, as well as to avoid the challenges caused by the large state/action dimensions. Consequently, the traffic signal control agent is more likely to gain a good understanding of the overall environment dynamics. Moreover, the proposed EGU-RL model contains fewer parameters and is end-to-end trainable. Our experimental results show the effectiveness of our model working in complex multi-interaction scenarios. Performance achieved by the proposed method surpasses multiple comparing methods in most cases.

3.3 Background

Recently, traffic optimisation has been extensively discussed. Traditional methods use fixed rules and detector-based methods to control the traffic lights. Detector-based methods (also referred to as actuated methods) leverage detectors placed underground with predefined rules to control traffic signals. Thus, they have more adaptability when comparing to fix-rule methods. Recently, many learning-based and data-driven approaches, i.e., evolutionary algorithms, have been adopted to effectively adapt traffic dynamics. The combination of learning-based methods and predefined control frameworks are also proposed by researchers, such as the auction-based methods Covell, Baluja, and Sukthankar, 2015; Schepperle and Böhm, 2008. The work in Baluja, Covell, and Sukthankar, 2017 incorporates an auction-based model with Next-Ascent Stochastic Hill-climbing (NASH) optimisation to control traffic lights.

The control of traffic lights can be modelled as a Markov Decision Process (MDP) Ritcher, 2007. As such, we can exploit reinforcement learning. A reinforcement learning agent takes the observation of current traffic conditions as its states. Following certain policies, the agent decides the next round of traffic signals to switch as its actions. Then, corresponding rewards of taking an action will be returned, after which the agent updates its model accordingly. The agent will always try to maximise the accumulated expected reward as much as possible. The aforementioned process forms a closed loop that enables the agent to learn the "correct" action distribution under a particular circumstance. With great progress achieved by deep learning techniques, many reinforcement learning algorithms combined with deep neural networks have been proposed, i.e., Deep-Q-Network Mnih et al., 2015, Actor-Critic Mnih et al., 2016 and Proximal Policy optimisation (PPO) Schulman et al., 2017; Heess et al., 2017, leading to significantly improved results. They can generally be categorised into value-based methods and policy-based methods. The application of deep reinforcement learning on traffic optimisation has also been adopted by Liang et al., 2018; Genders and Razavi, 2016; Richter, Aberdeen, and Yu, 2007; Richter, 2006; Mannion, Duggan, and Howley, 2016. Tong et al. Pham et al., 2013 presented a preliminary comparison between reinforcement learning approaches and distributed constraint optimisation approaches for traffic signal control.

However, for multi-intersection control, because of the high complexity of modelling this problem, experimental settings are usually different among existing works. Most works simplify the situation to single or few junctions, which is largely different from real situations, rendering these approaches less useful for real-world applications.

Moreover, these methods need to encode vast states and exponentially increased action spaces, leading to an inefficient learning procedure. Some works incorporate multi-agents methods Arel et al., 2010; Liu et al., 2018; Wang et al., 2020; Chu et al., 2019; Wu et al., 2020 to alleviate this issue. In practice, tuning hyper-parameters of a DRL model is not a trivial work and it is even harder to tune that of a multi-agent model.

If without effective communication between agents, a multi-agent system may suffer from unstable training easily and falls into sub-optimal solutions. This has been theoretically proven by research in game theory, among which the prisoner's dilemma is a typical example. Authors of Nishi et al., 2018 combined graph convolutional networks with reinforcement learning. The policies of each intersection are learned simultaneously but independently with an individual network per junction. It is worth noting that this work simplifies the environment to six intersections. It remains unclear how sensitive the results presented in Nishi et al., 2018 with respect to a large number of parameters, as multiple isolated agents do not have communications to each other. It also increases the risk of overfitting due to its large number of parameters. In Wei et al., 2019, a graph attention network was proposed for the traffic optimisation task. Nevertheless, multiple intersections are processed individually and separately which increases the difficulties of the model optimisation process due to the model is updated for every individual traffic signal.

Here, we propose a novel and strong baseline model to exploit relations among junctions by encoding their spatial and distance information with edge-weighted graph convolutional networks. Additionally, we introduce a unified structure decoder to jointly model multiple intersections and avoid the issue of exponentially increased action dimension explosion. The proposed method can not only outperform existing methods, but also significantly reduce the number of parameters.

On the settings aspect, Auction-based methods Covell, Baluja, and Sukthankar, 2015, the work in Schepperle and Böhm, 2008 and Baluja, Covell, and Sukthankar, 2017 attempted to minimise average travel time or total travel time of vehicles. The work in Nishi et al., 2018 adopts a position matrix to map the locations and the speeds of vehicles within one intersection as the state, the duration change of different phases as its action space and waiting time of vehicles as the model's reward. In Guo et al., 2019, individual vehicle's position and speed have also been used as the state. However, the reward is defined to minimise the total queue length of incoming lanes. The work in Mousavi, Schukat, and Howley, 2017 uses the raw images or snapshots of a graphical simulator as the input state. Besides the different state and reward designs, experiments only on synthetic data were conducted by most of the existing models. Thus, the settings and data provided by existing works are vastly inconsistent. Towards this end, we carefully design our settings and propose a new dataset that contains one synthetic map and several real maps from the city of Manhattan and Suzhou along with real traffic data.

3.4 Settings and Dataset

Traffic light signals consist of multiple phases. The number of traffic light phases is an indicator of how many traffic streams are allowed to occupy the road in one round. For instance, in practice four-phase traffic signals are commonly seen traffic signals

for crossroads. The intuitive illustration of a four-phase traffic signal is shown as Fig. 3.1 (taking right-side driving for example). Phase "1" and "3" represent left-turn of N-S and E-W directions respectively. Phase "2" and "4" stand for straight/right-turn of N-S and E-W directions respectively. In each round, phases are turned from "1" to "4" sequentially to allow different traffic streams to pass the junction without collision.

For one traffic light, each phase controls one or several non-conflicting lanes. In each phase, "green", "yellow" and "red" signals are flashed orderly for controlled lanes. While the phase of a traffic light is changing, some "red" lanes will be switched to "green". At the same time, some other lanes are switched from "green" to "red", during which "yellow" signals are inserted as intermediate phases for warning purpose. The "yellow" signals delay the effect of phase switching that complicates the problem even more.

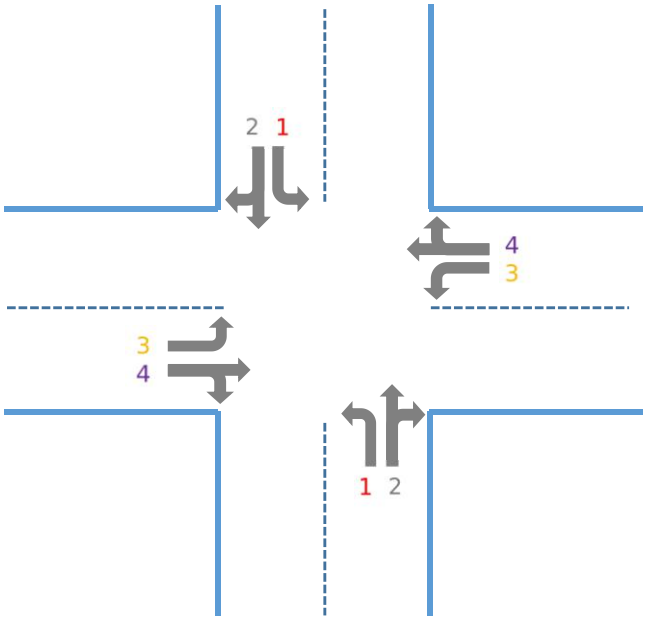

FIGURE 3.1. An example of traffic signals (right-side driving). As shown in the figure, the numbers represent different phases. In each round, phases are switched from "1" to "4" sequentially to allow different traffic streams to pass the junction without collision.

3.4.1 Formulation as a Markov Decision Process

The aforementioned traffic control task can be modelled as a Markov Decision Process (MDP). A typical Markov Decision Process can be described as a four-tuple $\langle S, A, R, P \rangle$, where S stands for the set of partially observed states. A is the set of all possible actions. R represents the collection of the rewards received by taking certain actions. P means the transition probability set among different states. The traffic optimisation problem formulation is described in detail subsequently.

State Space

To realistically simulate the environment, vehicles' starting/stopping acceleration and drivers' reaction time are taken into consideration. The goal of state spaces design is to encode observations of agents as comprehensive, practical and simplified as possible. Some existing works treat raw images as states Mousavi, Schukat, and Howley, 2017. Instead of raw images, different states have been designed in our settings, making model training more efficient.

The number of waiting vehicle state S_{num} is the concatenation of each single lane waiting vehicle number, which is shown as Eq. equation 3.1 and Eq. equation 3.2. It counts the number of vehicles if their velocity is below a certain threshold.

$$S_{num} = \{S^1_{num}, S^2_{num}, ..., S^m_{num}\} \tag{3.1}$$

$$S^j_{num} = \sum_i veh^t_i, \forall veh^t_i \in \text{VEH}_t, \text{ if } v(veh^t_i) \leq \text{Thresh}, \tag{3.2}$$

where m is the total lane number. S^j_{num} represents number of waiting vehicle for lane j. Thresh is a vehicle speed threshold. VEH_t is the set of vehicles within controlled lanes at time t. veh_i represents individual vehicle. $v(x)$ is the function to retrieve the speed of a vehicle.

However, S_{num} fails to take the overall waiting time of vehicles into account. Thus, we calculate the total vehicle's accumulative waiting time S_{wt} as well from an orthogonal angle. Similar to the number of waiting vehicle state, the total accumulative waiting time state S_{wt} is the concatenation of accumulative waiting time of individual lanes:

$$S_{wt} = \{S^1_{wt}, S^2_{wt}, ..., S^m_{wt}\} \tag{3.3}$$

$$S^j_{wt} = \sum_i \text{WT}(veh^t_i), \forall veh^t_i \in \text{VEH}_t \tag{3.4}$$

where $\text{WT}(veh^t_i)$ is the function to retrieve vehicles' accumulative waiting time. Thus, the state at time step t is the concatenation of S_{num} and S_{wt}:

$$S_t = \{S_{num}, S_{wt}\} \tag{3.5}$$

In this case, both the number and accumulated time of waiting vehicles are now taken into account. It empowers the model to treat vehicles with different waiting time differently. A typical example of traffic optimisation scenario is shown as Fig. 3.2.

Action Space

As described in the previous section, the number of traffic light phases is an important indicator to show how many traffic streams are allowed to occupy the road. Instead of designing a hard constraint, i.e., phase switch sequence and duration, we set each

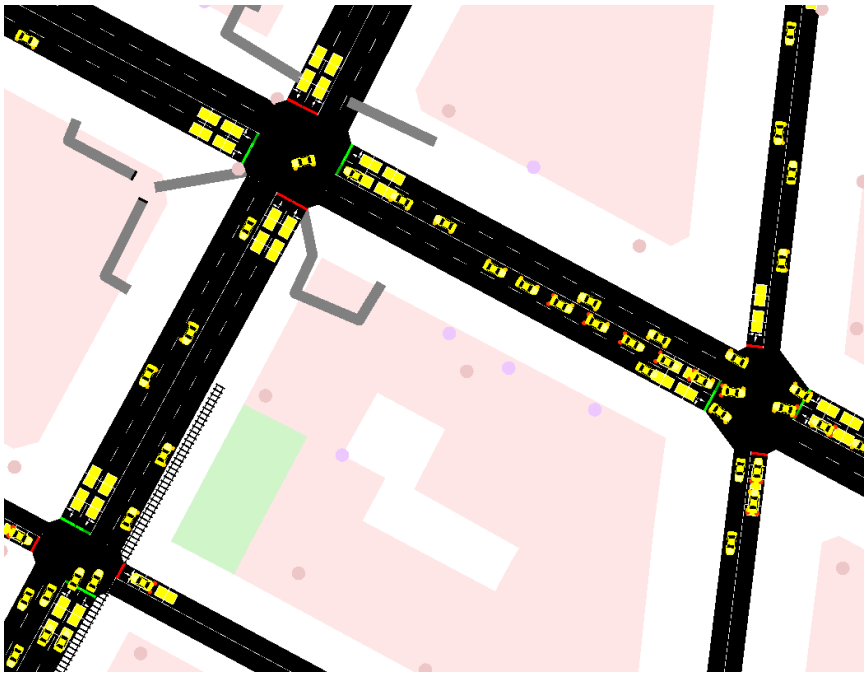

FIGURE 3.2. An example of traffic optimisation scenario. Based on the traffic scenarios, we design two states: (1) total number of waiting cars and (2) total vehicle's accumulative waiting time.

phases to be selected as actions. In this case, the controller is able to switch between phases freely to maximise the learning ability of models. More concretely, for a four-phase traffic signal controller, its actions space contains four elements which are corresponded to four different phases. If the initial phase is NS-Right-Turn (phase 1 in Fig. 3.1) and the controller tends to switch to Left Turning (phase 2 in Fig. 3.1), an action $1 \rightarrow 2$ is required.

The one-hot vector is the most commonly adopted encoding method. However, if we would like to control multiple traffic signals with a central agent, one-hot action vector representation is no longer an ideal solution. Because the phase combinations of different traffic lights are required to be encoded into one unified action space to ensure there is no conflicts and ambiguity, which incurs an exponentially increased action space.

In the design of state and action space, we propose an "group agent" strategy. In this strategy, intersections close to each other will be grouped together and they are controlled by a shared agent. We call it grouped multi-agent reinforcement learning model. With this strategy, actions can still be encoded as one-hot vectors since the number of traffic lights within a group is not significantly huge. For each action a satisfying $\forall a \epsilon A$. In term of the agent action space design, we have also proposed a unified structure decoder to output actions of all traffic lights at the same time, which has different action space definition and will be elaborated in detail in section IV.

Reward Design

Reward is the feedback from the environment to indicate the effectiveness of performed actions. In our settings, we define the total accumulative waiting time improvement as an instant reward at time step t, which is denoted as:

$$R_{wt} = \sum_i \mathrm{WT}(veh_i^{t-1}) - \sum_j \mathrm{WT}(veh_j^t) \tag{3.6}$$

Intuitively, if a "correct" action is performed, the total accumulative waiting time from the last step will be reduced significantly, so a large R_{wt} will be received; otherwise, a small R_{wt} will be returned.

Evaluation Metric

In the proposed traffic optimisation setting, the total accumulative waiting time cost is adopted as our evaluation metric to measure different algorithms. We define the total accumulative waiting time cost at each step t as:

$$Cost_{wt} = \sum_i \mathrm{WT}(veh_i^t), \forall veh_i^t \in \mathrm{VEH}_t \tag{3.7}$$

As stated in the previous section, $\mathrm{WT}(veh_i^t)$ also represents the accumulative waiting time of a certain car within control.

Eq. equation 3.7 reflects our life experience: if an appropriate action is perform, the accumulative waiting time cost at time step t is going to drop. Accordingly, a small $Cost_{wt}$ will be returned. Otherwise, the model will receive a larger accumulative waiting time cost. Therefore, the goal of our task is to minimise the total accumulative waiting time cost. If the cost is minimised, it means that a promising model is learned.

3.4.2 Data

In this subsection, we present the details of the data, including two parts: (1) traffic maps and traffic flow data. (2) statistics of these data.

Maps and Traffic Data

'Env' is a synthetic map with fifteen intersections. 'Man1' and 'Man2' are real traffic maps of Manhattan ares New York and each of them contains twenty-two intersections. 'Suzhou1' is a ten-intersection real map with real traffic flows collected from Suzhou China. 'Suzhou2' is a real scene from Suzhou as well with twelve intersections.

In our simulation, the aforementioned waiting car number state is retrieved from road sensors. Additionally, two detect loops are placed at the first five and ten meters respectively of each lane to retrieve the accumulative waiting time state. All of our

experiments are trained and evaluated in five different scenes. In the map Suzhou1, the real traffic flow was generated by tracking the from points and to destinations of real vehicles. Besides Suzhou1, three different traffic flows with different busy conditions are generated to keep data diversity. The details of our maps and data are shown in Table 3.1.

TABLE 3.1. Maps and traffic data. 'Env' is a synthetic scene with fifteen intersections. 'Man1' and 'Man2' are scenes of Manhattan blocks of New York and each of them contains twenty-two intersections. 'Suzhou1' is a ten-intersection real map with real-world traffic flows collected from Suzhou, China. 'Suzhou2' is a scene from the blocks of Suzhou with twelve intersections. 'Synth' represents the synthetic traffic data.

Scenes	Env	Man1	Man2	Suzhou1	Suzhou2
Maps	Synth	Real	Real	Real	Real
Traffic	Synth	Synth	Synth	Real	Synth
Inters Num	15	22	22	10	12

Simulation and Statistics of Data

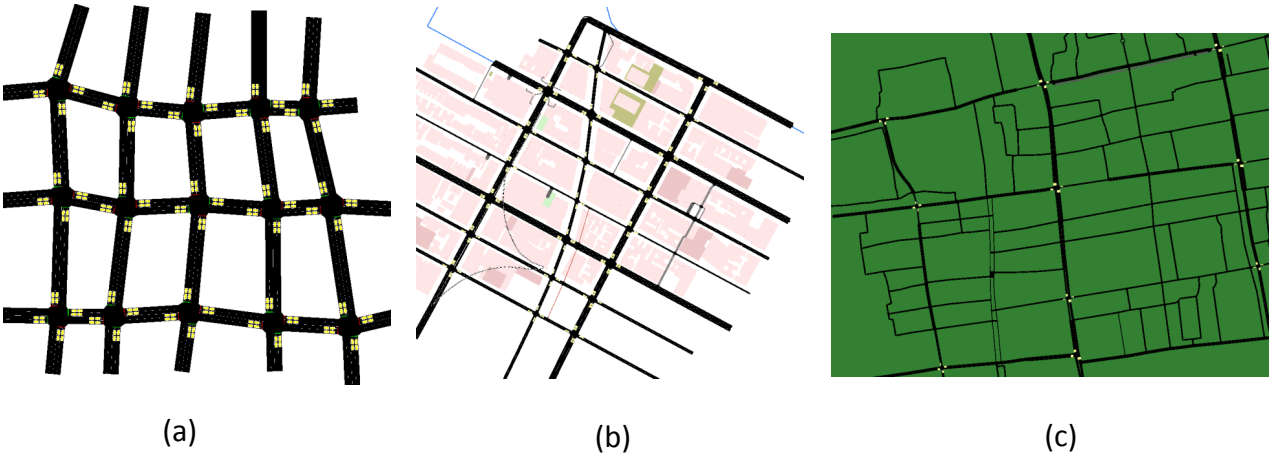

FIGURE 3.3. Some maps of the proposed dataset. In the figure, the (a) is the synthetic Env map; (b) is the Manhattan2 map and (c) is the Suzhou1 map.

Compared with existing works, more intersections have been included in our data, ranging from ten to twenty-two intersections. Some maps of the proposed dataset can be found in Fig. 3.3. Simulation of Urban MObility (SUMO) Lopez et al., 2018 has been used as our simulator throughout our experiments. In order to simulate closely to real situations, vehicles' starting/stopping acceleration and drivers' reaction time have been considered as well. Settings can be found in Table 3.2.

We split our traffic scenario into 1000 simulation steps for training. For the testing phase, we evaluate different models on seen simulation steps and further extend it to unseen ones for generalisation evaluation.

TABLE 3.2. Our simulation setting. 'Suzhou1' is a ten-intersection real map with real-world traffic flows dynamics collected from Suzhou China. For other scenes, in order to have a simulation close to real scenarios, vehicles' starting/stopping acceleration and drivers' reaction time (the time delays of drivers from observing state changes to take actions) and other important factors have been modelled.

Settings	Values
Starting Acceleration	2.0 m/s^2
Stopping Acceleration	4.5 m/s^2
Driver Reaction Time	0.8 s
Vehicle Length	4.5 m
Vehicle Gap	1.5 m
Speed Deviation	0.2

W.r.t. the generation of traffic flows. The real traffic flow was generated by tracking the from points and to destinations of real vehicles. The other synthetic traffic flows were generation with SUMO. We generated vehicles with equidistant departure times and period equals to 0.8, 1.0 and 1.2 to form the three flows of a scene. The weight of fringe edges is set to 10. The number of attempts for finding a trip which meets the distance constraints is set to 100.

3.5 EGU-RL Model

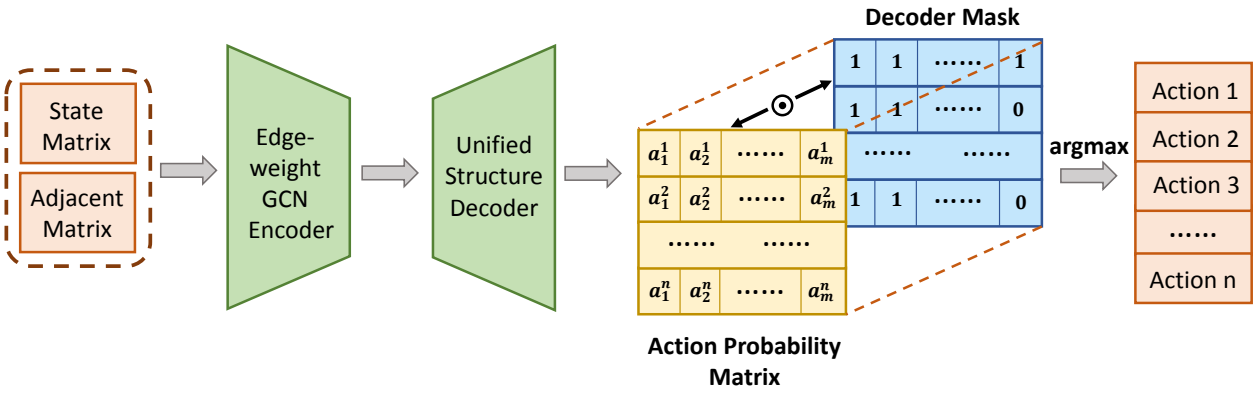

FIGURE 3.4. The proposed EGU-RL Model. The figure represents our proposed EGU-RL model with the Edge-weighted Graph Convolutional Encoder and the Unified Structure Decoder.

The proposed edge-weighted graph convolutional and unified structural Reinforcement Learning (EGU-RL) model contains two complementary parts: (1) An edge-weighted graph convolutional encoder to excavate positional relations among junctions. (2) A unified structure decoder to integrate the control of multiple intersections into the controller. The model structure is shown in Fig. 5.1.

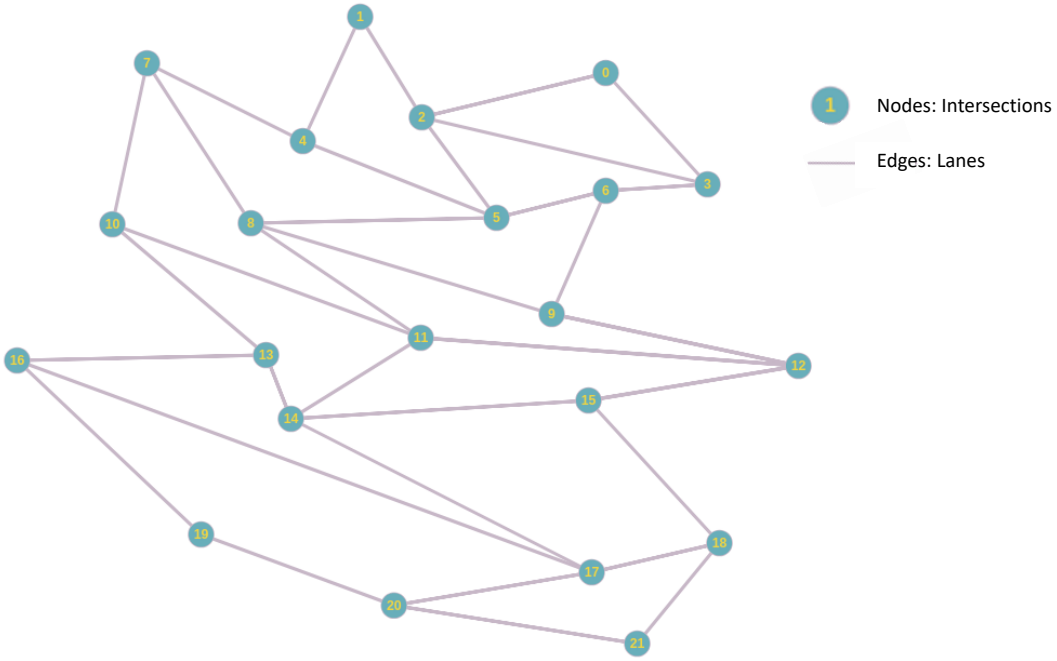

FIGURE 3.5. An Instance of graph model, which can be processed by Graph Convolutional Networks, to represent Junctions of several blocks in Manhattan. Each node represents an intersection and each edge represents a lane connection between two nodes.

3.5.1 Edge-weighted Graph Convolutional Encoder

Convolutional Neural Networks (CNNs) have achieved tremendous successes in most computer vision tasks. However, Standard CNN techniques are mostly designed to deal with euclidean data with grid-like structures Bronstein et al., 2017. However, for the traffic optimisation task, the spatial relations of intersections are graphs that makes the problem to be non-euclidean. Graph Neural Networks Kipf and Welling, 2016; Scarselli et al., 2009; Battaglia et al., 2018 (GNNs) offer a new point of view with a more generic framework to encode these topology node relations. Thus we further exploit cross-intersection relations using edge-weighted graph convolutional networks by taking the spatial and distance information into consideration.

For layer-wise representations, the idea of how the Graph Convolutional Networks work in the multi-intersection traffic optimisation problem is shown as below:

$$F^{(l+1)} = f(F^{(l)}, A), l \geq 1 \tag{3.8}$$

where $F^{(l)}$ is the feature map of layer l, A represents the adjacent matrix of all junctions. As shown above, the network structure sends the current layer's features and adjacent matrix into the network to produce features of the next layer. $F^{(0)} = X$, and X is the original input of the network.

Following the previous work Kipf and Welling, 2016, we use graph convolutional nets

combined with reinforcement learning to solve this optimisation problem. Additionally, we take a further step to propose an edge-weighted graph convolutional encoder to assign different weights to edges according to lane length. Because not only to know the topological relations among junctions is significant, but also to perceive the distance is crucial for the model optimisation process. It offers more information of edge importance than raw edge relations.

$$f(F^{(l)}, A) = \sigma(\widehat{A}F^{(l)}W^{(l)}) \tag{3.9}$$

$$\widehat{A} = normal(A + I) \tag{3.10}$$

where $W^{(l)}$ is the weights of layer l and σ here represents the activation function in our proposed model. The adjacent matrix A is defined with edge-weighted elements to attend on edges of different lengths. In this case, the elements within it is not constrained to 0 or 1. More specifically, the edge weights are the quotient of the maximum lane length divided by each lane length. Intuitively, the shorter of the length, the larger weight will be assigned. In terms of $\widehat{A}$, we follow the Graph Convolutional Networks to further normalise them row-wisely to alleviate node weights imbalance, and then add it to a identity matrix I, shown as Eq. equation 3.10

3.5.2 Unified Structure Decoder

In multi-intersection traffic optimisation, if intersections are close to each other, it is desired to control them as groups instead of treating them as separated ones. It is because grouped controllers are able to perceive more information and take appropriate actions by jointly modelling traffic flows as a whole. Otherwise, if each traffic light acts isolatedly, it may lead to sub-optimal results in dealing with urban-level traffic dynamics. Thus, we propose a grouped multi-agent reinforcement learning controller to group close-related intersections for action dimension reduction. For example, if three four-phase traffic signals are grouped, action space can be presented by 64 bits with the one-hot encoding, which is exponentially increasing with the addition of controlled junctions.

Based on that, we further propose a unified structure decoder for multi-intersection control to decoder actions of multiple traffic signals in a unified manner. With this strategy, agents are able to plan and control large-scale traffic nets comprehensively since there is no internal friction to coordinate different agents. Intuitively, instead of using individual one-hot vectors for an agent as actions, the unified structure decoder is able to control all intersections with a m by n action probability matrix, where n denotes the number of intersections and m denotes the phase number of each signal (e.g., $n = 4$ for a four-phase traffic signal). In the case of the phase numbers of various traffic lights are different, the action probability matrix is multiplied with a masked matrix to ensure each signal is controlled properly.

By doing so, the vast multiple signal action size can be reduced dramatically. The state is the concatenation of all controlled intersection observations. Thus, all junctions are taken into considerations. When using one-hot action encoding, the action space of three four-phase traffic lights is 64 bits ($4\times4\times4$, which is exponentially increasing with the addition of controlled junctions). However, with the proposed unified structure decoder, the outputted action is a 3×4 action matrix, which is 12 bits $(4 + 4 + 4)$. This is because we output them at the same time as an action matrix with the help of unified structure decoder. Thus, the model equipped with the unified structure decoder is able to deal with the dimensionality curse of exponentially increased action size. It enables EGU-RL model to control complex traffic flows with significant less parameters.

For the unified structure decoder, to let the agent have more comprehensive observations, we introduce a new state S_P to consider current phases of all the other traffic signals.

$$S_P = \{cur\_phase_1, cur\_phase_2, ..., cur\_phase_m\} \tag{3.11}$$

S_P is the concatenation of current phases of all traffic lights. m denotes the number of intersections within control. So the hybrid state space becomes:

$$S = \{S_N, S_T, S_P\} \tag{3.12}$$

3.5.3 Optimisation of the EGU-RL Model with Reforcement Learning

In the aspect of reinforcement learning, rewards indicate the effectiveness of performed actions. At time t, the discount cumulative reward R_t is:

$$\overline{R_t} = E[r_t + \lambda r_{t+1} + ...] = E[\sum_{i=0}^{\infty} \lambda^i r_{t+i}] \tag{3.13}$$

where r_t defines reward at time t, λ is the reward discount factor.

It is impractical to use the total travel time as a cost function since it cannot be calculated immediately at every time step which will dramatically slow down the model convergence. Therefore, we adopt the Temporal difference (TD) updating strategy instead of episode-wise updating strategy for the reinforcement learning learner to update the model. It achieves less training variance and is more data-efficient by combining with DQN experience replay.

Beside the total accumulative waiting time improvement reward R_{wt} stated in the previous section, in order to encourage the agent not switch phases frequently, we introduce another novel unchanged reward:

$$R_{uc} = \sum_i l_i, \forall l_i \epsilon L, \text{ if } l_i \text{ is unchanged} \tag{3.14}$$

where L represents the set of all traffic lights and l_i denotes each traffic light. Intuitively, the more unchanged traffic signals, the more rewards will be returned.

Therefore the total hybrid reward function is:

$$R_t = R_{wt} + \alpha R_{uc}, \tag{3.15}$$

where α is a trade-off factor between two rewards. To simplify the model optimisation process, we do not heuristically set any other constraints, i.e. minimum or maximum duration of each phase.

For optimisation algorithms, we choose Double Deep Q-Networks (DQN) Van Hasselt, Guez, and Silver, 2016 as our RL algorithms. Experience replay Schaul et al., 2015 and fix-target network techniques have also been used to improve the training efficiency and stability. During training, we update our model in the temporal difference (TD) manner rather than episode-wise. With the help of experience replay, the DQN-based EGU-RL model is transferred to an off-policy RL method. Mathematically, we have:

$$Q(s, a, \theta') = Q(s, a, \theta) + \beta[r_t + \gamma \max_{a'} Q(s', a', \phi) - Q(s, a, \theta)] \tag{3.16}$$

where θ', θ and ϕ are different learnable weights. β is the learning rate. r_t is the instant reward for time t defined by Eq. equation 3.15 and $Q(\cdot)$ represents Q networks.

3.6 Experiments

In this section, we report experimental details and results. Besides the EGU-RL model, the competing methods include fix-rules controller, random controller, auction-based model Baluja, Covell, and Sukthankar, 2017, single-agent for single-junction multi-agent reinforcement learning model (MARL-s) to control one junction with individual agent, and grouped multi-agent reinforcement learning model (MARL-g) that controls a group of traffic signals with one agent. As a competing method, the grouped multi-agent reinforcement learning model is also proposed in this chapter to group spatial-closed intersections for better control. We carry out extensive experiments to study the optimal agent configuration. We empirically find that as the number of

isolated agents increases, the performance drops significantly and generalises poorly on the unseen data. In contrast, our proposed EGU-RL model is able to process large-scale junctions with a jointly modelled agent and achieves promising results. All models are trained and evaluated on various maps of Suzhou, Manhattan and synthetic scenes.

3.6.1 Experimental Details

The experiments are conducted on the platform with an Intel Xeon CPU E5-2680 and 128GB RAM. To better simulate the real scenes, many simulation factors have been considered. Those settings are shown in Table 3.2. In the experiments, all models are implemented with PyTorch Paszke et al., 2017 and Simulation of Urban MObility (SUMO) simulator v0.32.0 Lopez et al., 2018. The proposed EGU-RL model and other competing methods are trained for 500 episodes and each episode contains 1000 simulation steps. For all DQN-based methods, experience replay Schaul et al., 2015 and fix target network Van Hasselt, Guez, and Silver, 2016 techniques are used. The target network for target-Q-value retrieving has an identical structure as the inference network and the parameters of the target network will be overridden every 100 steps. For EGU-RL method, the network structure is simple. It consists of 2 hidden layers. After obtaining and concatenating observations of each junction, the state matrix is inputted to a layer with 128 neurons. Then, the outputted features will be fed to a 64-neuron layer and then output an action probability matrix. In the grouped multi-agents reinforcement learning (MARL-g) model, each agent controls three close-neighboured intersections. The learning rate of each algorithm is configured as 10^{-3}. During training, the DQN reward discount factor $\gamma = 0.9$ is set. The agent's new action exploration rate is set to 0.5 initially, and reduce to its 0.99999 each learning step until reaches 0.01. Additionally, the buffer size for experience replay and batch size are set to be 2000 and 150, respectively. We keep all aforementioned hyper-parameters unchanged throughout our experiments. All of the models in our experiments are trained from scratch for a fair comparison.

3.6.2 Model Performance

The performance of different models are presented in Table 4.2. Each model was tested on seen 1000 simulation steps for each scene. The vehicles' total accumulative waiting time cost is used as evaluation metric (the lower the better). In the table, "flow1", "flow2", "flow3" are the traffic flows generated with different tempo of vehicles passing by and "flow" is the real traffic flow.

From Table 4.2, within 13 scenes of seen environments, the EGU-RL method achieves 10 best scores. "Auction" represents the model proposed in the paper Baluja, Covell, and Sukthankar, 2017. It combines the auction-based model with the Next-Ascent Stochastic Hill-climbing (NASH) algorithm to control traffic lights, which is a greedy controller.

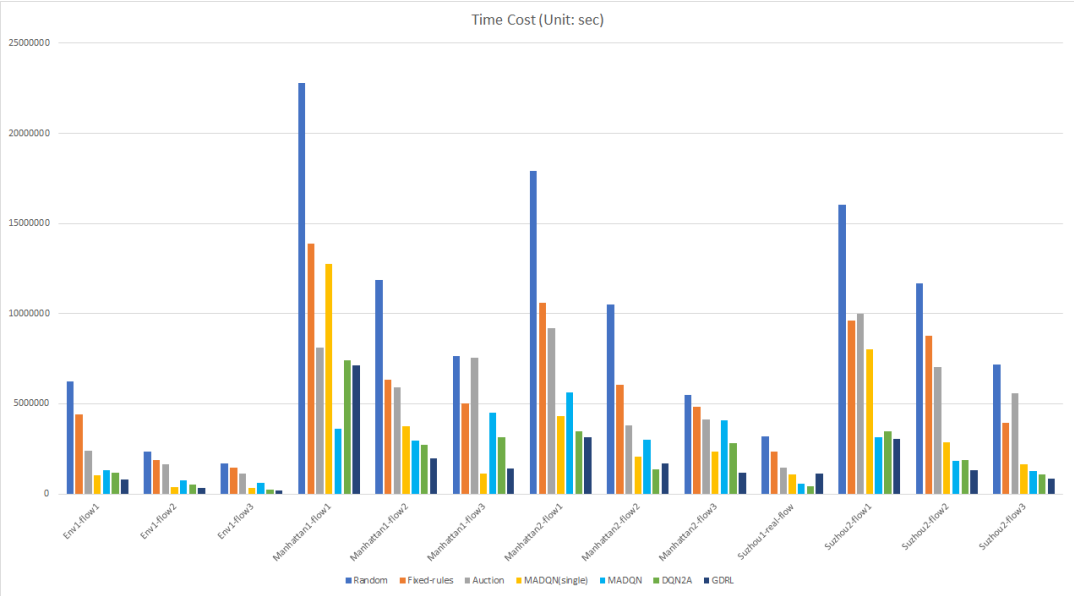

FIGURE 3.6. Performance comparison of different models. EGU-RL can achieve the best accumulative waiting time cost among all the other methods in most cases.

TABLE 3.3. Cross-model accumulated waiting time consumption performance comparison in seen environments. The unit here is hours. The less accumulated waiting time consumed, the better performance of the model has. The EGU-RL method achieves the best results among all the other methods in most cases. The best results for each row are in bold.

Scenes	Random	Fix	Auction	MARL-s	MARL-g	EGU-RL
Env-flow1	1729	1230	661	283	362	**222**
Env-flow2	654	526	461	108	210	**86**
Env-flow3	475	409	319	95	175	**57**
Man1-flow1	6332	3850	2249	3537	**1007**	1978
Man1-flow2	3298	1753	1646	1036	816	**549**
Man1-flow3	2127	1396	2099	**313**	1251	395
Man2-flow1	4972	2943	2558	1197	1560	**875**
Man2-flow2	2919	1675	1052	573	829	**469**
Man2-flow3	1530	1342	1148	658	1134	**321**
Suzhou1-flow	883	657	409	302	**151**	315
Suzhou2-flow1	4461	2671	2781	2231	870	**843**
Suzhou2-flow2	3242	2438	1959	794	514	**368**
Suzhou2-flow3	1989	1090	1551	456	354	**239**

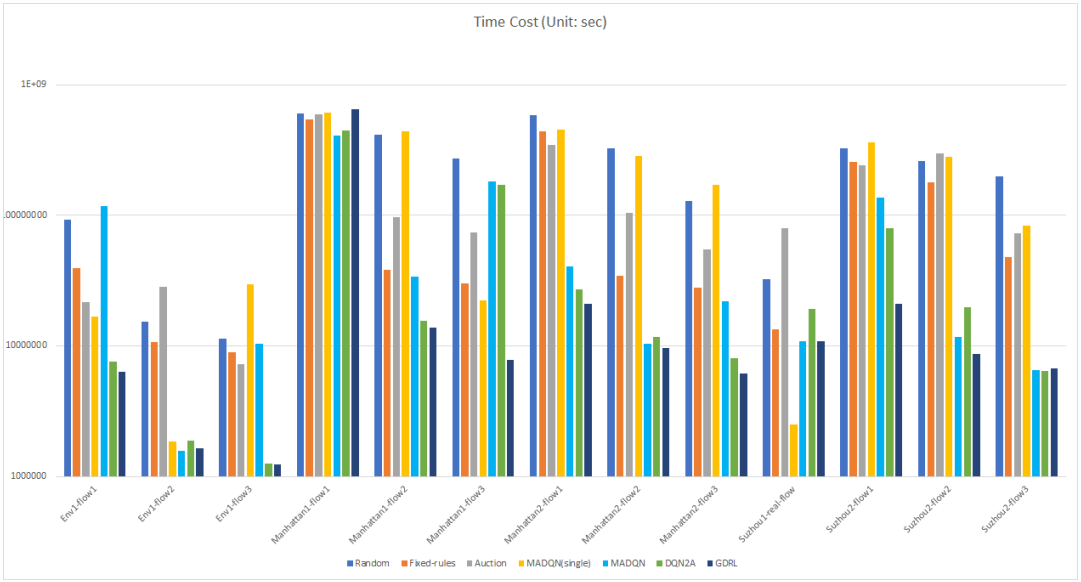

FIGURE 3.7. Generalisation ability comparison of different models. The EGU-RL has the better generalisation ability compared to other methods in most cases (log y-axis has been used to avoid large difference between different columns).

However, the results show the inferior performance of the greedy controller for multi-intersection control. The greedy controller does not receive ideal results probably caused by the high complexity and the fast-changing dynamics of traffic flows. The EGU-RL model can reduce the total accumulative waiting time by up to more than 80% when compared to the fixed-scheduled controller and surpass other methods by a large margin in most cases, especially in the environments, such as Env and Suzhou2, with complex positional relationship of intersections. More specifically, EGU-RL reduces 21.6% and 28.4% total vehicle's accumulative waiting time compared to the second-best model in Env-flow1 and Suzhou2-flow2 respectively. In the table, our model outperforms the MARL-s model. This is due to a naive application of a multi-agent system that without effective communication and cooperation may significantly influence the model performance, even perform worse than a centralized agent. On the contrary, the proposed centralized agent does not have internal consumption and it thus can perform well under some situation, especially under the scenarios that the traffic lights are close to each other, e.g., Env-flow1 and Man2-flow1 scenarios.

We have conducted two sets of experiments by settings and not settings the length of phases. We empirically found that vehicles will not wait infinitely if we do not set the phases length. This may be because if a vehicle waits for a long time at one intersection, the cost will be high and the agent will try to reduce the cost and release the vehicle.

We extend our model evaluation to 5000 unseen simulation steps. The total vehicles' accumulative waiting time cost is used as the evaluation metric. The generalisation evaluation is shown in Table 3.4.

TABLE 3.4. Accumulated waiting time consumption comparison of different models in unseen environments. Similar to seen scenarios, the unit here is hours. The less accumulated waiting time consumed, the better performance of the model has. In the generalisation test, the EGU-RL method also achieves the best results in most cases. The unit here is hours.

Scenes	Random	Fix	Auction	MARL-s	MARL-g	EGU-RL
Env-flow1	25644	10916	6018	4620	32572	**1745**
Env-flow2	4268	2961	7839	515	**433**	456
Env-flow3	3171	2494	2003	8187	2869	**343**
Man1-flow1	167626	149491	163488	170034	**113034**	180230
Man1-flow2	115670	10556	26936	121190	9369	**3857**
Man1-flow3	75594	8322	20403	6204	50263	**2165**
Man2-flow1	161138	121285	95819	126198	11189	**5791**
Man2-flow2	89933	9579	28790	78835	2897	**2684**
Man2-flow3	35788	7764	15137	47274	6102	**1702**
Suzhou1-flow	9024	3716	22083	**697**	3017	2995
Suzhou2-flow1	90027	71567	67021	100385	37727	**5788**
Suzhou2-flow2	72071	49926	82210	78221	3263	**2393**
Suzhou2-flow3	54948	13270	20084	22987	**1820**	1868

From Table 3.4, we can clearly see that the proposed EGU-RL model shows a better generalisation ability to unseen environments compared to other methods. Similar with seen experiments, within 13 scenes of seen environments, the EGU-RL method achieves 9 best scores. More specifically, EGU-RL reduces 65.1% and 26.7% total vehicle's accumulative waiting time compared to the second-best model in Man1-flow3 and Suzhou2-flow2 respectively. Our model can achieve better performance particularly in complex environments. This is because EGU-RL model takes junction topological relations into account and contains less parameters. Nevertheless, the naive multi-agents systems overfit easily since it is difficult for isolated agents cooperate with each other lacking in-depth communication.

3.6.3 Parameter Numbers

The EGU-RL method can not only achieve promising results, but also has fewer parameters. The comparison of parameter numbers is shown in Table 3.5, where EGU-RL\EGE represents EGU-RL model without edge-weighted graph convolutional encoder.

TABLE 3.5. The parameter number of different models. EGU-RL has the least parameters among all the other algorithms in different scenes.

Scenes	Env	Man1	Man2	Suzhou1	Suzhou2
MARL-s	412.92k	542.38k	587.57k	274.61k	327.64k
MARL-g	279.04k	349.09k	405.85k	162.04k	218.59k
EGU-RL\EGE	169.33k	185.52k	230.32k	114.26k	132.96k
EGU-RL	**27.78k**	**27.78k**	**27.78k**	**39.94k**	**30.34k**

A model may have different parameter numbers in different environments due to various input and output sizes. From Table 3.5, MARL-s has the most parameters since an individual agent controls each intersections, followed by grouped multi-agent reinforcement learning model and w/o EGE model. Our proposed EGU-RL method has the least parameters and saves more than 85% of parameters compared to MARL-s method.

3.6.4 Inference Time

What is more, we also conduct experiments to measure the actual inference time of the EGU-RL model. We fetch the inference time by averaging the results of 10 times on the platform of Intel i7-8700 CPU (without GPU acceleration). The results are shown in the Table 3.6.

TABLE 3.6. The inference time of EGU-RL model in different scenes. The unit is in ms.

Scenes	Env	Man1	Man2	Suzhou1	Suzhou2
Inference Time	0.396	0.451	0.467	0.387	0.391

As shown in the table, in the synthesis 15-intersection simulated environment Env, the inference time is 0.396 ms. In Man1 and Man2, due to 22 intersections are contained in each scene, the inference time is slightly slow, which are 0.451 ms and 0.467 ms respectively. In Suzhou1 real map/flows environment, our EGU-RL model is able to predict the next action in 0.387 ms. In Suzhou2, the inference time is 0.391 ms. The results of the table show the general robustness of the model and there is no latency issue. Thus, the proposed EGU-RL model is capable of dealing with large-scale traffic data.

3.6.5 Ablation Study

To evaluate the effectiveness of individual components in our model, we further conduct ablation studies. Table 3.7 shows the performance influence of the edge-weighted graph convolutional encoder and the unified structure decoder in seen simulations.

From Table 3.7 we can see that, without USD or EGE, the results experience a significant drop. “w/o USD” model only outperforms other two methods on Suzhou2-flow2 and Suzhou2-flow3 scenes, while beaten by its counterparts in other scenes. “w/o EGE” only performs well on two scenes. It is because our proposed EGU-RL model is able to not only excavate multi-intersection spatial relations with edge-weighted graph convolutional encoder, but also jointly model all controlled junctions with the unified structure decoder.

We also carry out experiments on 5000 unseen environments to validate the generalisation ability of different components, shown as Table 3.8. “w/o USD” performs poorly on most of the scenes with complex traffic flows. However, “w/o EGE” and EGU-RL

perform relatively better, since the unified structure decoder integrates all intersections to predict actions in a more comprehensive manner. Models equipped with the unified structure decoder has remarkably fewer parameters and will not overfit easily. The edge-weighted graph convolutional encoder takes crucial cross-intersection relations into account to further boost the performance. It is easier for our network to learn complex interactions of multiple intersections because the actions are jointly modelled.

The cost convergence comparison is shown in Figure 3.8. The EGU-RL algorithm surpasses the MARL-g method and w/o EGE method and achieves the lowest cost of total accumulative waiting time in the end. Moreover, our proposed EGU-RL algorithm shows much stabler training curves than the other two models. The corresponding rewards obtained by different models are shown in Fig. 3.9. Similarly, our proposed EGU-RL model surpasses MARL-g and w/o EGE models in the end by a large margin.

We have also compared the reward obtained by EGU-RL with hybrid reward and with only total accumulated waiting time reward. Empirically, the reward trade-off factor α is set to 1.0. Figure 3.10 suggests that with hybrid reward, the proposed model is able to receive much more reward. Consequently, it represents the effectiveness of hybrid reward to practically control the frequency of traffic signals.

We further conduct extra experiments on vehicles' stop times. In SUMO, the vehicles' stop time returns the consecutive time in where this vehicle was standing (voluntary stopping is excluded). Therefore, we sum up the stop waiting time of each vehicle as the evaluation. The results are listed in the table 3.9

Similar to evaluation under accumulative waiting time of vehicles, the stop time optimised by the proposed EGU-RL is significantly lower than the competing methods, including Fix-schedule scheme, Auction-base method Baluja, Covell, and Sukthankar, 2017 and Single-agent for single-junction strategy. The EGU-RL model reduces more than 75% of vehicle stop time compared with Fix-schedule scheme under the Man2-flow1 scenario with busy traffic flow. It also reduces 50% of vehicle stop time when compared with MARL-s.

3.6.6 EGU-RL with other Reinforcement Learning algorithms

Our proposed EGU-RL model is flexible to combine with different reinforcement learning algorithms. Actor-Critic (AC) algorithm with EGE and USD further proves the point. The results are shown in Table 3.10.

In Table 3.10, both EGU-RL-DQN and EGU-RL-AC models are able to achieve promising results on Env-flow1 environment.

3.7 Conclusion

In this chapter, we have carefully designed settings for machine learning based traffic control. We have also proposed new datasets, consists of one synthetic map and multiple real maps along with real traffic flows. We have proposed a novel and strong EGU-RL model to tackle multi-intersection traffic optimisation problems.

The proposed EGU-RL model is able to not only exploit multi-intersection spatial relations with an edge-weighted graph convolutional encoder, but also jointly models all controlled junctions with the unified structure decoder. The model integrates all junctions by outputting an action probability matrix with a unified structure decoder. By doing so, significantly fewer parameters are required to train and thus it reduces the overfitting risk. It facilitates networks to learn complex interactions of different intersections because the actions are jointly modelled. Experiments show that the performance achieved by the proposed model surpasses existing methods and the proposed model is more effective and practical in real traffic control scenes due to its simplicity.

TABLE 3.7. The performance of ablation variants in seen environments. In the table, "EGE" denotes edge-weighted graph convolutional encoder and "USD" represents unified structure decoder. "w/o −" means the EGU-RL variant that removes the the corresponding module from EGU-RL.

Scenes	w/o USD	w/o EGE	w/o EW	EGU-RL
Env-flow1	387	332	275	**222**
Env-flow2	131	142	107	**86**
Env-flow3	262	71	105	**57**
Man1-flow1	2308	2053	**1563**	1978
Man1-flow2	4107	760	793	**549**
Man1-flow3	534	880	475	**395**
Man2-flow1	1461	968	**745**	875
Man2-flow2	667	**381**	446	469
Man2-flow3	365	777	340	**321**
Suzhou1-flow	344	**120**	451	315
Suzhou2-flow1	1485	963	864	**843**
Suzhou2-flow2	**267**	521	509	368
Suzhou2-flow3	**193**	303	255	239

TABLE 3.8. The performance of ablation variants in unseen environments.

Scenes	w/o USD	w/o EGE	w/o EW	EGU-RL
Env-flow1	5296	2109	**1717**	1745
Env-flow2	659	523	615	**456**
Env-flow3	3451	346	634	**343**
Man1-flow1	151606	**123722**	143547	180230
Man1-flow2	138592	4337	4410	**3857**
Man1-flow3	32345	47575	2675	**2165**
Man2-flow1	133800	7495	7371	**5791**
Man2-flow2	4617	3247	2745	**2684**
Man2-flow3	1938	2226	2078	**1702**
Suzhou1-flow	**2225**	5358	6546	2995
Suzhou2-flow1	36794	22073	**4903**	5788
Suzhou2-flow2	**2039**	5519	3705	2393
Suzhou2-flow3	**1267**	1789	3109	1868

TABLE 3.9. Cross-model vehicles' stop time consumption comparison in 1000 steps. The unit here is hours. The less vehicles' stop time consumed, the better performance of the model has. The EGU-RL method achieves the best results among all the other methods in most cases.

Scenes	Fix	Auction	MARL-s	EGU-RL
Man2-flow1	1571	1049	712	**390**

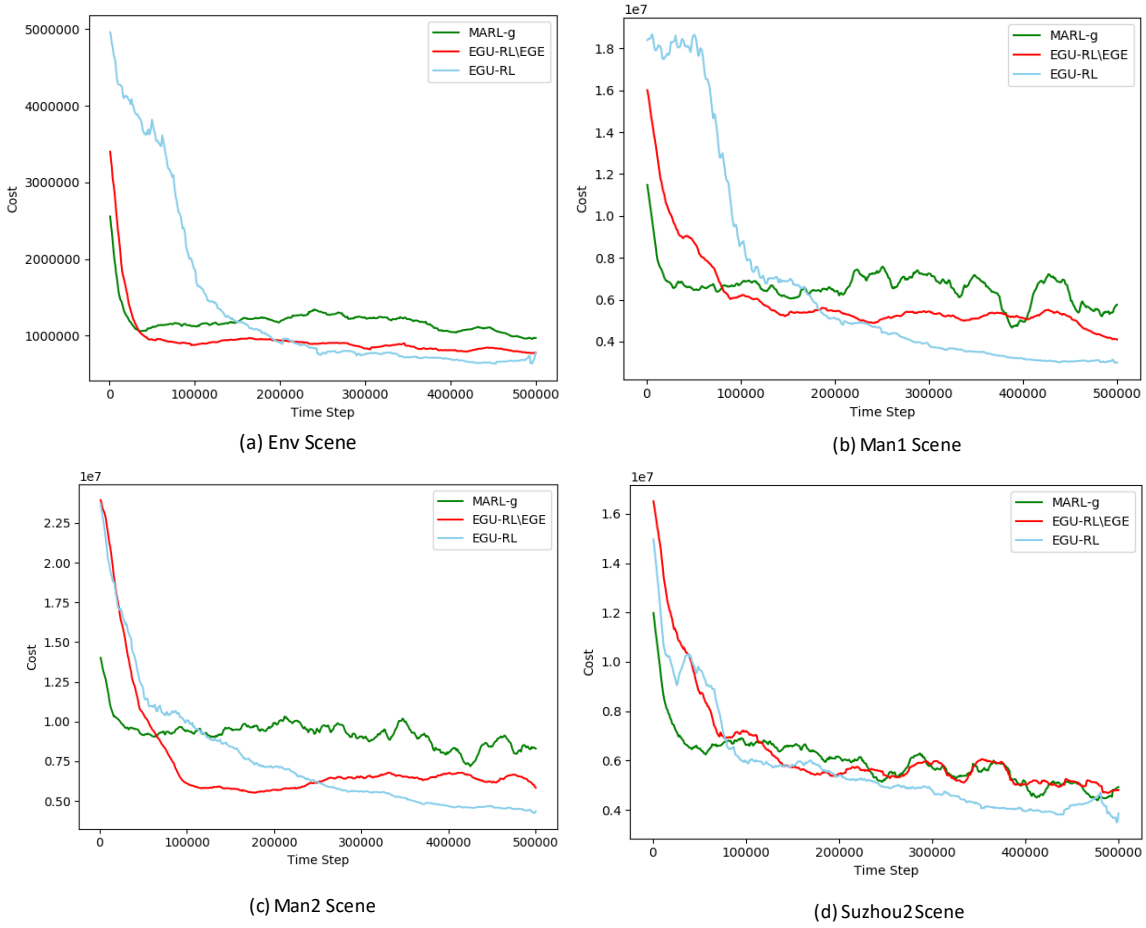

FIGURE 3.8. Convergence comparison of different models on total accumulative waiting time cost (the lower the better) under different scenes. The EGU-RL algorithm surpasses MARL-g and w/o EGE algorithms and achieved the lowest accumulative waiting time costs in the end.

TABLE 3.10. The Actor-Critic (AC) algorithm with EGE and USD are shown as EGU-RL-AC. Performance comparison of EGU-RL-DQN and EGU-RL-AC on different environments.

Scenes	EGU-RL-DQN	EGU-RL-AC
Env-flow1	**222**	269
Env-flow2	**86**	96
Env-flow3	57	**57**
Man1-flow1	1978	**1730**
Man1-flow2	**549**	555
Man1-flow3	395	**355**
Man2-flow1	875	**726**
Man2-flow2	**469**	488
Man2-flow3	321	**289**
Suzhou1-flow	315	**167**
Suzhou2-flow1	843	**793**
Suzhou2-flow2	**368**	382
Suzhou2-flow3	**239**	268

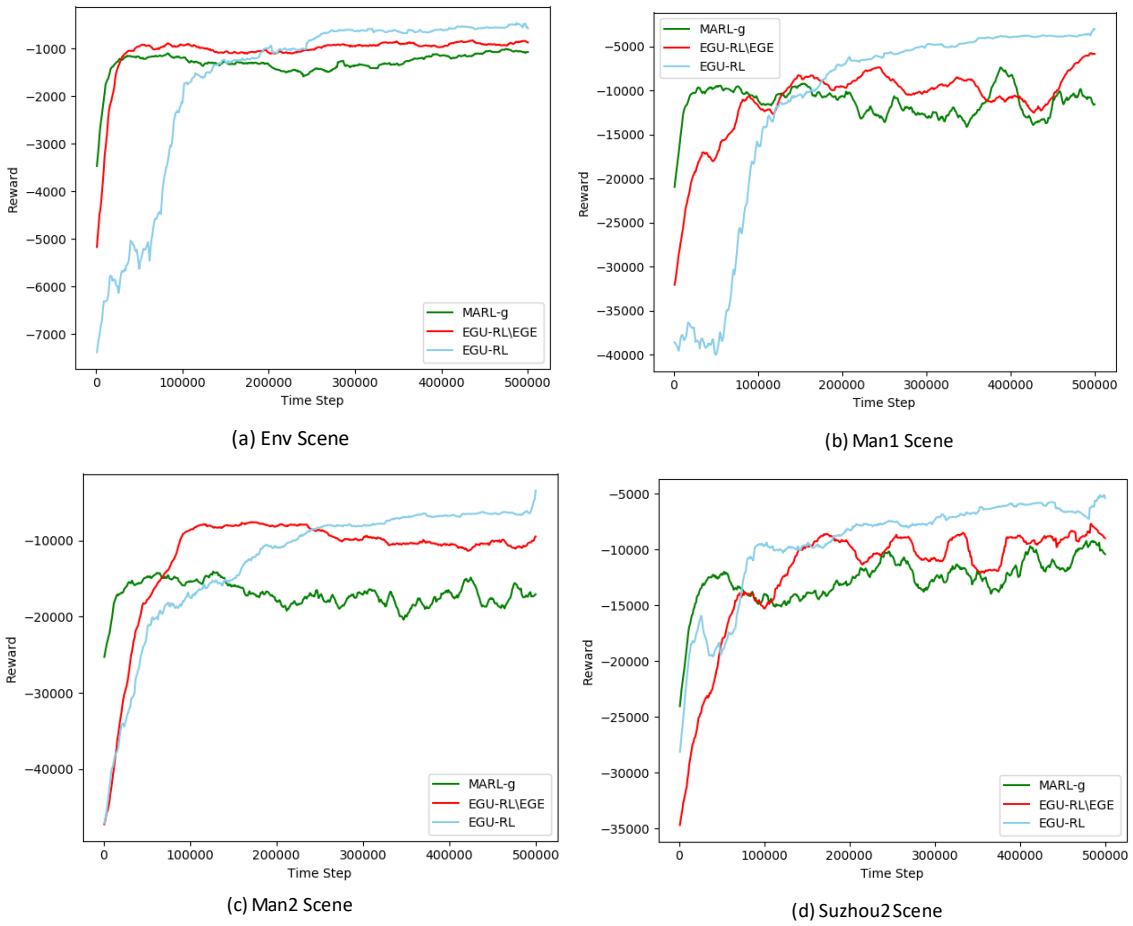

FIGURE 3.9. Convergence comparison of different models on total rewards (the higher the better) under different scenes. EGU-RL algorithm surpasses MARL-g and w/o EGE algorithms and gets the highest reward in the end.

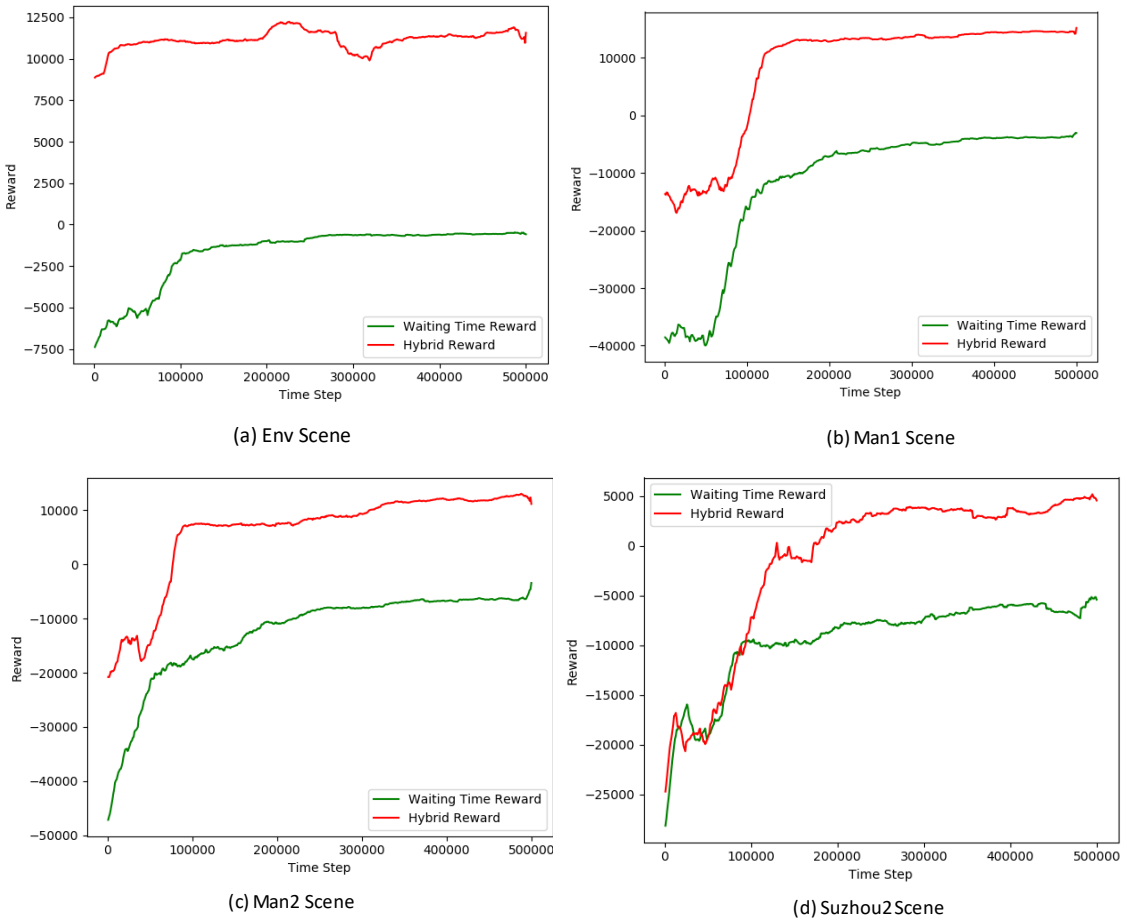

FIGURE 3.10. Comparison of EGU-RL with hybrid reward and with only waiting time reward under different scenes. The proposed algorithm can obtain higher reward with hybrid reward, which suggests the effectiveness of hybrid reward to practically control the frequency of traffic signals.

Chapter 4

Soft Expert Reward Learning for Vision-and-Language Navigation

4.1 Overview

Vision-and-Language Navigation (VLN) requires an agent to find a specified spot in an unseen environment by following natural language instructions. Dominant methods based on supervised learning clone expert's behaviours and thus perform better on seen environments, while showing restricted performance on unseen ones. Reinforcement Learning (RL) based models show better generalisation ability but have issues as well, requiring large amounts of manual reward engineering is one of which. In this chapter, we introduce a Soft Expert Reward Learning (SERL) model to overcome the reward engineering design and generalisation problems of the VLN task. Our proposed method consists of two complementary components: Soft Expert Distillation (SED) module encourages agents to behave like an expert as much as possible, but in a soft fashion; Self Perceiving (SP) module targets at pushing the agent towards the final destination as fast as possible. Empirically, we evaluate our model on the VLN seen, unseen and test splits and the model outperforms the state-of-the-art methods on most of the evaluation metrics.

4.2 Introduction

Vision-and-Language Navigation (VLN) tasks Anderson et al., 2018b define a comprehensive problem: an embodied agent is placed at a spot in a photo-realistic house and the agent is called to navigate to a specific spot based on given natural language instructions. Rising research interests have been put into the VLN since multi-modal data are involved. One of the biggest challenges for this task is to ask an agent to perform appropriate actions in an unseen environment. This in turn requires the agent to learn human behaviours to understand and explore the scene, instead of memorising it.

Current VLN models Anderson et al., 2018b; Fried et al., 2018; Ke et al., 2019; Ma et al., 2019a; Ma et al., 2019b rely much on behavioural cloning (BC) that treats

expert behaviours as strong supervision signals. By doing this, it enables the agents to gain better performance on seen scenarios, however the agents meet trouble on unseen environments due to the error accumulation. As stated in Reddy, Dragan, and Levine, 2019, teacher forcing models suffer from distribution shift issues because of the greediness of imitating demonstrated expert actions.

Some other works Tan, Yu, and Bansal, 2019; Wang et al., 2019b, instead, adopt reinforcement learning (RL) along with supervised learning methods intending to overcome the error accumulation issue caused by hard behavioural cloning. However, reward engineering in RL suffers issues: the reward functions designed at one environment/task may not generalise well to other scenarios; in many practical and complicated tasks, it is hard to define concrete reward functions as game scores. What is more, a hand-crafted reward is defined to target a certain functionality, it thus inevitably incurs lacking comprehensive consideration of the system dynamics. The design of a reward function requires careful manual tuning and it also suffers generalisation problems due to environment-oriented reward designing, which may affect the model performance while inference.

In this chapter, we propose a Soft Expert Reward Learning (SERL) model to address the above issues. Our proposed method consists of two orthogonal parts: the Soft Expert Distillation (SED) module that portrays the expert data distribution by distilling knowledge from a random projection space and a Self Perceiving (SP) module that encourages agents to reach the goal as soon as possible. For the SED module, intuitively, a higher reward should be assigned to an agent who takes an action "close" to its expert. To measure the similarity continuously, a density function was adopted to reflect this process in a soft manner rather than leveraging behaviour cloning directly. This density function is implemented to calculate the similarity between observation-action pairs of the expert and the agent in a randomly projected space, by doing which it transforms the expert behaviour into a soft reward signal for the reinforcement learning branch. For the Self Perceiving (SP) module, our model first predicts the schedule to the target location and then utilises the predicted schedule information as an additional reward. As a result, the agent can perceive its current schedule and use it to further push itself forward toward the goal.

The two newly designed reward modules work complementarily: the Soft Expert Distillation (SED) reward encourages agents to behave as experts, but the soften behaviour-imitation process makes it more robust; Self Perceiving (SP) module targets at pushing the agents towards the final destination by introducing the current schedule information as another intrinsic reward signal. In summary, this chapter makes the following three main contributions.

- We propose a Soft Expert Distillation (SED) formulation, which is very simple yet offers a highly effective reward signal for obtaining expressive navigational

ability. The SED reward encourages the agent to have a better alignment with its expert in a soft manner.

- We introduce another complementary reward signal with aforementioned SED reward termed as Self Perceiving reward that can help the agent use the current schedule information to push itself to reach the destination as soon as possible.
- As a result, we show our instantiated model termed as SERL that enables better performance than current state-of-the-art competing methods in both validation unseen and test unseen set of VLN Room-to-Room dataset Anderson et al., 2018b.

4.3 Background

4.3.1 Vision-and-Language Navigation

In order to gain promising performance on Vision-and-Language (VLN) Anderson et al., 2018b task, numerous methods have been proposed, as listed in Table 4.1. Many existing works adopt supervised learning and behaviour cloning based methods. Seq2seq Anderson et al., 2018b model is the most naive baseline that utilises an LSTM-based sequence-to-sequence architecture with attention mechanism to predict the next action. Speaker-Follower Fried et al., 2018 model designs a language model ("speaker") to learn the relationship between visual and language information, as well as a policy network ("follower") to take actions based on multi-modal inputs. It uses "speaker" to synthesise new instructions for data augmentation and help the policy network to select routes. Ke et al., 2019 claims its proposed FAST model is able to balance local and global signals while exploring an unobserved environment. It enables the agent act greedily but allows the agent backtrack if necessary according to global signals. Ma et al., 2019a proposes a visual-language co-grounding framework named as self-monitoring model to better fuse the instructions and visual inputs. Building upon self-monitoring model, Ma et al., 2019b provides a strategy for the agent to retrieve and re-choose paths based on monitored progress.

Reinforcement learning Mnih et al., 2013; Schulman et al., 2017; Lillicrap et al., 2015 is another paradigm for parameter optimisation. Wang *et al.* Wang et al., 2019b propose a novel Reinforced Cross-modal Matching (RCM) via reinforcement learning to enforce cross-modal matching locally and globally along with imitation learning. In RCM model, an extrinsic reward measuring the reduced distance toward the target location after taking actions, as well as an intrinsic cross-modal matching reward between trajectories and instructions, are proposed. Most recently, Tan, Yu, and Bansal, 2019 introduces a novel environment dropout to drop features channel-wisely targeting at feature maps inconsistency issue through combining behaviour cloning and reinforcement learning.

TABLE 4.1. Performance Evaluation across different methods.

Methods	Behaviour Cloning	Reinforcement Learning	Reward Engineering	Reward Learning
Random Anderson et al., 2018b				
Seq2seq Anderson et al., 2018b	✓			
Speaker-Follower Fried et al., 2018	✓			
FAST Ke et al., 2019	✓			
Reinforced Cross-Modal Wang et al., 2019b	✓	✓	✓	
Self-Monitoring Ma et al., 2019a	✓			
Regretful Agent Ma et al., 2019b	✓			
EnvDrop Tan, Yu, and Bansal, 2019	✓	✓	✓	
SERL (Ours)	✓	✓	✓	✓

However, these approaches require either exact imitation of the expert demonstrations or careful reward designing. Behaviour cloning techniques unfortunately lead to error accumulation and further result in catastrophic failure while the agent is exploring unknown environments. Moreover, reward engineering requires careful manual tuning, which motivates us to propose SERL model to learn reward functions from the expert distribution directly.

4.3.2 Reward Learning

Reward engineering is commonly used to design reward functions for reinforcement learning algorithms. In conventional reinforcement learning tasks, such as playing Atari games Brockman et al., 2016, rewards are individually shaped by each game simulators. However, reward engineering has obvious drawbacks — the reward functions are designed targeting at different environments which is not generic. There are some methods have been proposed to solve this problem. Recently, Inverse reinforcement learning (IRL) Ng, Russell, et al., 2000 framework is proposed to extract reward functions from expert behaviours by updating both of the reward functions and the policy networks. Random Expert Distillation (RED) Wang et al., 2019a proposed an expert policy support estimation method to distil rewards from given expert trajectories. Generative Adversarial Imitation Learning (GAIL) Ho and Ermon, 2016 is also a recently proposed model which tries to bypass the reward function and learn experts behaviour directly with generative adversarial networks.

Comparing with the IRL and GAIL models, our proposed Soft Expert Distillation module learns expert demonstration data distribution directly by comparing the output similarity between a randomised network and a distillation network, rather than utilising iterative model updating and generative adversarial networks. The RED model designs state and action in relatively small spaces for the Mujoco environment Todorov, Erez, and Tassa, 2012 and its driving task; while we design our SED module in fundamentally different state and action spaces for navigation in photo-realistic Matterport3D environments. We are the first to introduce soft expert reward learning framework into Vision-and-Language task.

4.4 Soft Expert Reward Learning Model

4.4.1 Overview and Problem Definition

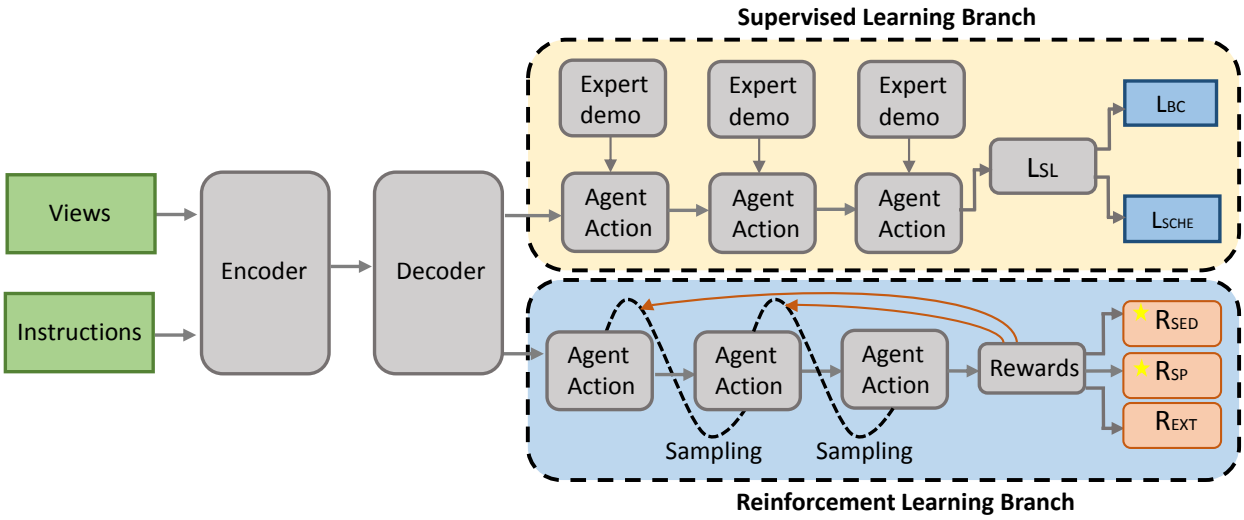

FIGURE 4.1. The proposed Soft Expert Reward Learning (SERL) framework. After getting the visual features and language features through the encoder, they are fed into the decoder to obtain the selected action a_t for time step t. The training process of SERL is divided into two parts: a supervised learning branch and a reinforcement learning branch. We introduce two novel rewards (marked with yellow stars in the figure): Soft Expert Distillation (SED) reward and Self Perceiving (SP) reward.

Vision-and-Language Navigation task requires an agent placed at a unknown photo-realistic house to understand multi-modal data comprehensively, so that the agent can navigate to the specified location. The multi-modal data includes natural image data and natural language instructions. More specifically, after an agent is spawn, at each time step t the observation of the agent consists of 36 images of panoramic views, denoted as $V_t = \{v_{t,1}, v_{t,2}, ..., v_{t,36}\}$. The navigable views $N_t = \{n_{t,1}, n_{t,2}, ..., n_{t,k}, n_{t,k+1}\}$ are given as well, where k denotes the maximum number of navigable viewpoints and $n_{t,k+1}$ represents "stay" action. A m words length instruction is given which is denoted as $X = \{x_1, x_2, ..., x_m\}$. Based on the visual and language information, actions at each time a_t will be selected and eventually a trajectory $\tau = \{a_1, a_2, ..., a_T\}$ is formed. The objective of VLN task is to find the optimal action a_t^* at each step to quickly reach the target location, while keep the trajectory τ as short as possible. Since Vision-and-Language Navigation task is a sequential decision problem, it can be modelled as a Markov Decision Process (MDP), which is noted as a four-element-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. $\mathcal{S}$ and $\mathcal{A}$ represent state and action sets relatively. $\mathcal{P}$ is the environment dynamics and it can be presented in the form $\mathcal{P}(s, s') = P(s'|s, a)$. $\mathcal{R}$ is the reward function.

In this chapter we introduce a Soft Expert Reward Learning model to distil reward function directly from expert demonstrations and soften the process of behaviour

cloning to alleviate the drawbacks from error accumulation. The structure of our model is illustrated through Figure 5.1. We follow a standard Encoder-Decoder paradigm. The encoder plays the role as a multi-modal data feature extractor to fetch the features from both visual images and language instructions. The decoder is a LSTM (long short-term memory) network with attention mechanism to predict actions according to the abovementioned two branches: the supervised learning branch helps the agent imitate the expert demonstration and perceive the current schedule to the target location; the reinforcement learning branch optimises the outputted action probability distribution from reinforcement learning aspects. The key difference of our proposed SERL model with previous models is that we proposed two novel intrinsic reward signals: Soft Expert Distillation reward R_{SED} encourages the agent to align with expert actions but in a soft fashion and Self Perceiving reward R_{SP} motivates the agent to reach the goal as fast as possible with predicted schedule information. In the following sections, we will first introduce the Encoder-Decoder structure and then introduce the two reward functions.

4.4.2 Encoder-Decoder Structure

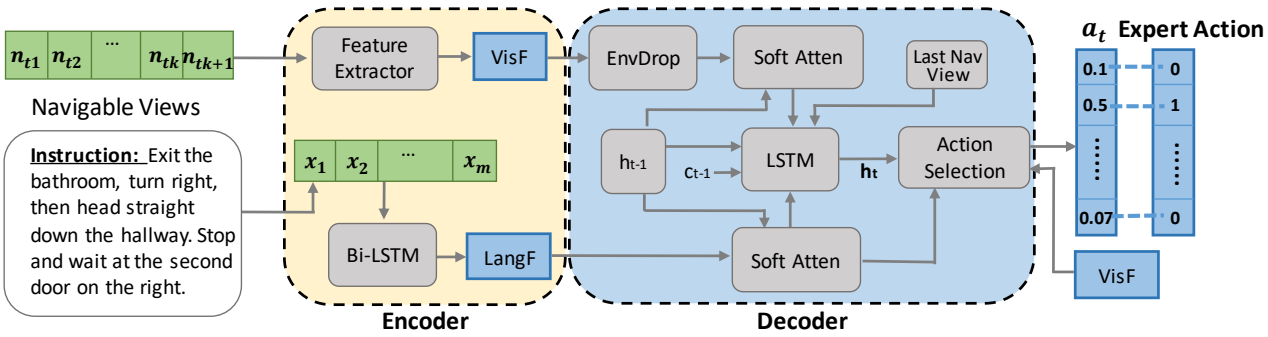

FIGURE 4.2. Encoder-Decoder Structure of Soft Expert Reward Learning (SERL) framework. After fetching the visual and language features from the encoder, the multi-modal features are fed into the decoder to obtain cross-modal attentions. Finally, actions will be chosen according to the attentive features.

Encoder-Decoder structure (as shown in 4.2) is adopted as the main structure of our method. Natural image data and natural language instructions are inputted to an encoder to extract corresponding features maps. Following the paper Ma et al., 2019a; Tan, Yu, and Bansal, 2019, we extract ResNet He et al., 2016 features of the navigable views concatenated with the orientation as the visual features $VisF_t$. We then use a Bi-Directional Long Short-Term Memory (Bi-LSTM) to pull out language features $LangF_t$. The multi-modal features are fed into a decoder to output the next action probability vectors later on.

Encoder:

On the encoder side, after pre-extracting ResNet features of different views, the feature maps of each navigable view $n_{t,i}$ is attached with an orientation tag $(\cos\gamma_{t,i}, \sin\gamma_{t,i}, \cos\varphi_{t,i}, \sin\varphi_{t,i})$ to form the visual feature $VisF_t$:

$$VisF_t = concat(resnet(n_{t,i}), (\cos\gamma_{t,i}, \sin\gamma_{t,i}, \cos\varphi_{t,i}, \sin\varphi_{t,i})), \tag{4.1}$$

where $concat(.)$ is a concatenation function.

For the language perspective, after each word of the instruction is tokenised into a vector, the token vectors are fed into a Bi-LSTM network to extract the language features $LangF_t$. As Eqn. 4.2, formally we have

$$LangF_t = \{x'_1, x'_2, ..., x'_m\} = Bi\text{-}LSTM(\{x_1, x_2, ..., x_m\}), \tag{4.2}$$

where x'_i is the corresponding i-th encoded word tokenised by Bi-LSTM.

Decoder:

On the decoder side, after the visual feature $VisF_t$ and language features $LangF_t$ are formed, along with the last cross-modal hidden state h_{t-1}, they are fed into soft attention layers to fetch the attentive visual and language features. Following the work Tan, Yu, and Bansal, 2019, the environment dropout is used on $VisF_t$ before feeding into soft attention layer to obtain feature-wise dropout for consistency in different views. Formally,

$$\widetilde{VisF}_t = Soft\text{-}Atten(EnvDrop(VisF_t), h_{t-1}), \tag{4.3}$$

$$\widetilde{LangF}_t = Soft\text{-}Atten(LangF_t, h_{t-1}). \tag{4.4}$$

Together with previous navigated view pre^v_{t-1}, last cross-modal hidden state h_{t-1}, cell state c_{t-1}, attentive visual and language features are fed into a LSTM layer to form the cross-modal hidden state h_t and cell state c_t at step t. This step is critical for the model to fuse the visual and language multi-modal signals to choose the action.

$$h_t, c_t = LSTM(h_{t-1}, c_{t-1}, pre^v_{t-1}, \widetilde{VisF}_t, \widetilde{LangF}_t). \tag{4.5}$$

The action probability distribution for the next step is calculated as:

$$p_t = softmax(fc(\widetilde{LangF}_t, drop(h_t)) \cdot VisF_t), \tag{4.6}$$

where $drop(.)$ represents a dropout function. The dot product $\cdot$ is used hereafter for matrix multiplication operation.

The decoder is connected to two branches: supervised learning branch and reinforcement learning branch. These two branches optimise the outputted action probability

distribution from two different learning paradigms. In this case, the total loss function is:

$$L = L_{SL} + L_{RL}. \tag{4.7}$$

SL Branch:

In the supervised learning branch, the cross-entropy loss between the predicted action logits and expert actions one-hot vector is calculated to force the agent to mimic its teacher's behaviours. This loss is termed as behaviour cloning loss L_{BC}. Following the work Ma et al., 2019a, besides the behaviour cloning loss, another loss to predict current schedule towards the goal is adopted. This loss is named as schedule loss L_{SCHE} working as an additional supervisory signal. Formally, the loss function for the supervised learning branch is:

$$L_{SL} = L_{BC} + L_{SCHE}. \tag{4.8}$$

where the behaviour cloning loss L_{BC} can be presented detailedly:

$$L_{BC} = -\sum_{i}^{n} y_{t,i}^{act} log(p_{t,i}), \tag{4.9}$$

where p_t and y_t^{act} are predicted action logits and expert actions one-hot vector at step t respectively.

To calculate the L_{SCHE}, the model ought to predict distance improvement ratio in advance at each step as its current schedule information. Then, L2 distance between predicted schedule and the genuine schedule is chosen as the loss function. Formally,

$$L_{SCHE} = (y_t^{sche} - V_t^{sche})^2, \tag{4.10}$$

where V_t^{sche} represents the predicted schedule which will be described in detail in the subsequent section and y_t^{sche} is the corresponding true schedule value.

RL Branch:

As the reinforcement learning branch shown in Figure 5.1, we adopt actor-critic algorithm Mnih et al., 2016 as our reinforcement learning method. For the reinforcement learning branch, the training loss L_{RL} can be formally represented as:

$$L_{RL} = \underbrace{\sum_{t} -log(p_t) * (\bar{R}_t - v(h_t))}_{actor\ loss} + \underbrace{\sum_{t} (\bar{R}_t - v(h_t))^2}_{critic\ loss}, \tag{4.11}$$

where $v(.)$ is the value function of critic. $\bar{R}_t$ represents the discounted reward for time step t and it can be formulated as:

$$\bar{R}_t = \bar{R_{t+1}} * \gamma + R_t, \tag{4.12}$$

in which the γ is the discount factor. The reward R_t is made up of three parts: an extrinsic reward R_{EXT} and another two complementary and newly proposed reward functions — Soft Expert Distillation (SED) reward R_{SED} and Self Perceiving reward R_{SP}. The total reward function thus can be formalised as:

$$R_t = \alpha R_{SED} + \beta R_{SP} + R_{EXT}, \tag{4.13}$$

where (1) SED reward R_{SED}, an automatically learnt reward function through aligning agent's behaviours to the provided expert demonstrations. (2) SP reward R_{SP}, a reward function comes from predicted schedule to encourage the agent to reach the goal as soon as possible. (3) The extrinsic reward R_{EXT} assigns the agent a positive reward, if the agent stops within three-meter from target or the agent reduces the distance to the goal; otherwise, a negative reward will be returned. α, β are the trade-off factors of SED reward and SP reward respectively. The details of individual proposed reward function will be revealed in the following sections.

4.4.3 Soft Expert Distillation

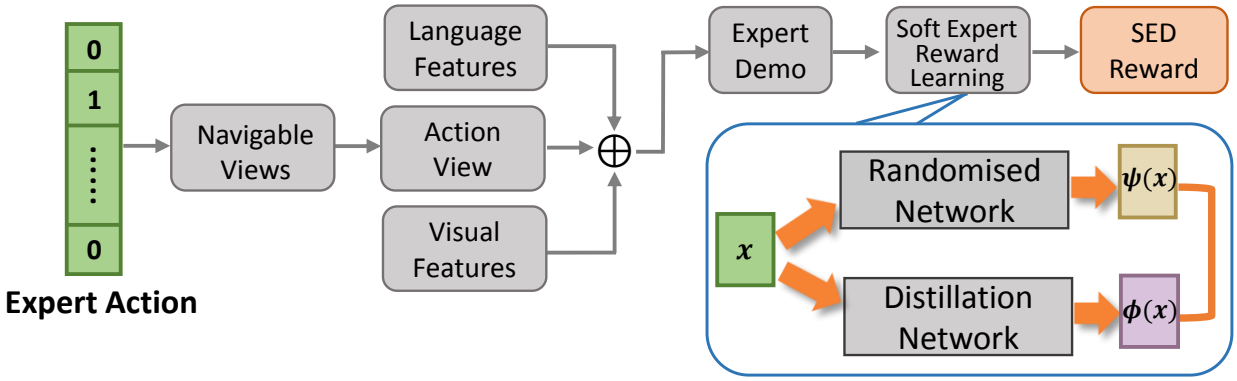

FIGURE 4.3. The Soft Expert Distillation networks structure. Given an expert demonstrated data point $x \in \mathbb{R}^N$, it is fed into a weight-fixed randomly initialised neural network $\psi(\mathbf{x})$; simultaneously, the data point x is inputted into a distillation network $\phi(\mathbf{x}; \theta)$ with different structure but same output dimensions with the parameters θ.

Inspired by the work Wang et al., 2019a, we propose to learn the reward function from inputted expert demonstration in Vision-and-Language Navigation task. We train a neural network to predict the output of a random-initialised but frozen network to distil the expert knowledge. The Soft Expert Distillation networks structure is shown in Figure 4.3. The key intuition behind this is: given a certain amount of random projection information, the representation learner is required to fit the structure of these given data points in the random projection space to achieve a similar projected distribution. The learning function is expected to predict relatively better where more expert data lays. In this case, a strong density function is formed. It models the likelihood of the agent performing a similar action with its expert in a situation through distillation. A higher prediction distance, which results in a low SED reward

in turn, will be assigned to unexpected observation-action pairs that differs from given expert demonstrations. Thus, a higher reward will be assigned to an agent who takes an action similar with its expert. This encapsulation of density function gives us another view of learning expert demonstrations directly other than Ng, Russell, et al., 2000 and Ho and Ermon, 2016.

Precisely, for a given expert demonstrated data point $x \in \mathbb{R}^N$, we first feed it into a weight-fixed and random-initialised neural network $\psi(\mathbf{x})$; at the same time the data point x is inputted into a distillation network $\phi(\mathbf{x};\theta)$ with different structure but same output dimensions. The data is projected to a M-dimensional new space by a representation learner $\phi : \mathbb{R}^N \mapsto \mathbb{R}^M$ with the parameters θ. We emphasise here, the function capacity of network ϕ is less than network ψ, by doing which can prevent overfitting. As we adopt L2 distance as our loss function, then we formulate the subsequent step as a prediction task and define a loss function as:

$$L_{sed}(\mathbf{x}) = (\phi(\mathbf{x};\theta), \psi(\mathbf{x}))^2 , \tag{4.14}$$

Empirically, both of ψ and ϕ are implemented by multi-layer perceptrons. $\psi : \mathbb{R}^N \mapsto \mathbb{R}^M$ plays the role of a random data mapping function to project points into a randomly projected space. By doing so, this loss offers a simple yet powerful supervisory signal for the distillation network to learn semantic-rich feature representations from given expert data processed by the random projection function ψ.

In order to distil the expert behaviour distribution, the data points are consist of expert's visual observation, language instructions and actions. The equation is formally shown as:

$$L^t_{sed} = (\phi(\{VisF_t, LangF_t, a_t\};\theta) - \psi(\{VisF_t, LangF_t, a_t\}))^2 . \tag{4.15}$$

The SED module preserves semantic-rich information w.r.t. distribution of expert demonstration for the representation learner. So the module is an ideal density function to measure the similarity of an agent's behaviour with the expert demonstration. Differ from the behaviour cloning process, it is formed in a soft manner. The SED intrinsic reward function is formally presented as:

$$R_{SED} = \begin{cases} +2, if Dis^{sed}_t <= thresh \\ -2, if Dis^{sed}_t > thresh \end{cases} \tag{4.16}$$

The L2 distance between $\phi(\{VisF_t, LangF_t, a_t\};\theta)$ and $\psi(\{VisF_t, LangF_t, a_t\})$ is denoted as Dis^{sed}_t. Intuitively, if Dis^{sed}_t is less than the threshold, it represents the current behaviour of the agent is similar with the expert distribution where a positive reward should be awarded; otherwise, a negative reward will be returned. In contrast, behaviour cloning based models encourage the agent to copy expert demonstrations

exactly; while our proposed soft expert distillation module learns the demonstrated behaviour in a soft manner by depicting the distribution of expert behaviours. In the case, the agent can retain the expert knowledge but will not suffer from the error accumulation problem. Thus, it increases the robustness of the model across various VLN environments.

4.4.4 Self Perceiving Reward

To perceive the schedule information towards the goal is crucial for the agent to complete the VLN task. A self perceiving module is designed to predict distance improvement ratio at each step as current schedule information of the agent. In order to utilise the information more adequately, we take one more step ahead by making use of this schedule information as another intrinsic reward—self perceiving reward. Formally, the self perceiving reward is calculated from:

$$C_{attn} = softmax(fc(h_{t-1}) \cdot LangF_t), \tag{4.17}$$

$$R_{SP} = V_t^{sche} = \sigma(fc(drop(\tanh(c_t) \odot \sigma(fc(h_{t-1}, \widetilde{VisF_t}))), C_{attn})), \tag{4.18}$$

where C_{attn} represents the language attention over different vocabularies within the instruction sentence. $\odot$ is the element-wise Hadamard product. Intuitively, the Self Perceiving reward indicates the predicted schedule information toward the destination. The more distance improvement ratio of the current action archived, the higher reward ought to be assigned. Moreover, this reward offers more information of distance change than raw distances. The more self perceiving reward the agent collected, the closer the agent believes to reach the target location.

4.5 Experiments

Following previous works Anderson et al., 2018b; Fried et al., 2018; Ke et al., 2019; Ma et al., 2019a; Ma et al., 2019b; Tan, Yu, and Bansal, 2019; Wang et al., 2019b, we evaluate our model on the Room-to-Room (R2R) dataset Anderson et al., 2018b for VLN task. Furthermore, we test our method on the VLN test server[1] Yadav et al., 2019 to validate the proposed Soft Expert Reward Learning Model. Ablation study is further conveyed to examine the contribution of each individual component of the model. The experimental results show the effectiveness of the proposed model.

[1]The VLN leaderboard address is https://evalai.cloudcv.org/web/challenges/challenge-page/97/leaderboard/270.

4.5.1 Experimental Setup

Evaluation Metrics. Currently, a variety of metrics are used to evaluate VLN models. We adopt the following metrics: Navigation Error (NE) is to measure the shortest path distance between the stopping position and the goal; Success Rate (SR) quantifies the rate of success if the agent can stop within three meters from the target; Oracle Success Rate (OSR) is the success percentage if the agent can stop at the closest point along its trajectory; the Success rate weighted by Path Length (SPL) Anderson et al., 2018a is also adopted to indicate the weighted SR.

Implementation Detail. Following Fried et al., 2018; Tan, Yu, and Bansal, 2019, we utilise the ResNet-152 model pre-trained on ImageNet to extract CNN features as visual inputs. Empirically, we set the M equal to 128, and set both of the reward trade-off factors α and β to 0.1. In Soft Expert Distillation networks, the randomised network is made up of two hidden linear layers with 512 and 256 neurons respectively; the distillation network has one hidden linear layers with 256 neurons. Between every two linear layers, both of the randomised network and the distillation network adopt leaky-relu as their activation function. To prevent overfitting, we early-stopped the training process of models according to the performance on the validation set. The Soft Expert Distillation module is not jointly trained with the rest of the model. This decoupling prevents performance unstableness during training and increase the robustness of the model.

4.5.2 Overall Performance

TABLE 4.2. Performance Evaluation across different methods. The first place of each column is bolded. All of the results are reported on models without beam search, except FAST Ke et al., 2019 model using a beam-search style strategy. The ↑ means that the higher the better; vice versa. The * sign represents data augmentation.

	Val Seen				Val Unseen				Test Unseen			
Methods	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑	NE ↓	SR ↑	OSR ↑	SPL ↑
Random	9.45	0.16	0.21	-	9.23	0.16	0.22	-	9.77	0.13	0.18	-
Seq2seq	6.01	0.39	0.53	-	7.81	0.22	0.28	-	7.85	0.20	0.27	0.18
Self-Monitoring	3.72	0.63	**0.75**	0.56	5.98	0.44	0.58	0.30	-	-	-	-
Regretful-Agent	3.69	0.65	0.72	**0.59**	5.36	0.48	**0.61**	0.37	-	-	-	-
EnvDrop	4.71	0.55	-	0.53	5.49	0.47	-	0.43	-	-	-	-
SERL (Ours)	**3.67**	**0.66**	0.71	0.58	**4.97**	**0.50**	0.59	**0.44**	**5.70**	**0.51**	**0.57**	**0.47**
Speaker-Follower*	3.36	0.66	0.74	-	6.62	0.36	0.45	-	6.62	0.35	-	0.28
RCM*	3.37	0.67	0.77	-	5.88	0.43	0.52	-	6.12	0.43	0.50	0.38
FAST*	-	-	-	-	4.97	**0.56**	-	0.43	**5.14**	**0.54**	-	0.41
Self-Monitoring*	3.22	0.67	**0.78**	0.58	5.52	0.45	0.56	0.32	5.67	0.48	0.59	0.35
Regretful-Agent*	3.23	**0.69**	0.77	0.63	5.32	0.50	0.59	0.41	5.69	0.48	0.56	0.40
EnvDrop*	3.99	0.62	-	0.59	5.22	0.52	-	**0.48**	5.23	0.51	0.59	0.47
EnvDrop-Our-Impl*	3.77	0.66	0.72	0.62	5.49	0.49	0.56	0.45	-	-	-	-
SERL* (Ours)	**3.20**	**0.69**	0.75	**0.64**	**4.74**	**0.56**	**0.65**	**0.48**	5.63	0.53	**0.61**	**0.49**

In this section, we convey the evaluation experiments on three individual sets, validation seen, validation unseen and test set, shown in table 4.2, to compare the effectiveness of our proposed soft expert reward learning model with other models. The comparison is split into two groups: models trained on non-augmented data and augmented data. Within twelve indicators of validation set and test set, we achieve ten best results on the non-augmented group and nine best results on the augmented group, which reveals the effectiveness of SERL model. More specifically, for the non-augmented group, on validation unseen set, our SERL model reduces the navigation error by 7%, increase the success rate by 4% and SPL by 2%. Our method also receives remarkable results on test unseen set. Similarly for the augmented group, on validation unseen set, it is clear that our model is the best performer. SERL model reduces the navigation error by 5% and gets 0.56 successful rate. Our model also increases 10% for the oracle successful rate and gets 0.48 SPL respectively compared to the second-best model. On the test unseen set, our SERL model can achieve performance better than, or comparably well to, the other competing methods in Table 4.2. When compared to the second-best model, the model increases 3% for the oracle successful rate and 4% SPL respectively. The FAST Ke et al., 2019 model applies a beam-search style strategy, thus it is expected to produce better successful rate (SR) but it leads to a relatively worse SPL.

4.5.3 Ablation Study

Ablation Study of Different Components Performance

This section examines the contribution of each component of SERL model. Different components are added to the baseline model. The ablation results are represented as Table 4.3. The results are shown on validation seen and unseen sets and the models are trained with the same data augmentation strategy. In the first column, SED represents our proposed soft expert distillation module, while SP is the self perceiving module. BS represents beam search setting. We check different components in the second column to examine each variant. Row model #1 shows the performance of the environment dropout methods that we implemented. From the table we can clearly find that when comparing to row #1, excluding the beam search setting on the validation unseen set, the model with SED module alone (method #2) achieves higher SR by 6% and increases SPL score from 0.45 to 0.48; the model with SP module alone (#3) receives better success rate as 0.53 from 0.49 and better SPL score as 0.46 from 0.45. This is because the SED module encourages the agent to have better alignment with expert trajectories, but in a soft way; the SP module pushes the agent to find the target location as fast as possible. The full SERL model (method #4) combines the advantages of individual module and it achieves 0.56 of successful rate and 0.48 of SPL, which outperforms other variants.

Additionally, beam search is another popular Vision-and-Language Navigation setting. In the beam search setting, the agents are given the chance to choose the trajectories

TABLE 4.3. Ablation study of different components in SERL model. We evaluate the results on validation seen set and validation unseen set. The best result are bolded.

Models	SED	SP	BS	Val Seen NE ↓	SR ↑	OSR ↑	SPL ↑	Val Unseen NE ↓	SR ↑	OSR ↑	SPL ↑
1				3.77	0.66	0.72	0.62	5.49	0.49	0.56	0.45
2	✓			3.67	0.66	0.74	0.63	5.10	0.52	0.58	**0.48**
3		✓		3.19	0.67	0.72	0.61	4.93	0.53	0.61	0.46
4	✓	✓		3.20	0.69	0.75	**0.64**	4.74	0.56	0.65	**0.48**
5	✓	✓	✓	**2.47**	**0.77**	**0.99**	0.02	**3.01**	**0.71**	**0.99**	0.02

with the highest success rate. In this case, it can further boost the success rate of our SERL model (method #5) to 0.77 on validation seen set and 0.71 on validation seen set. Moreover, SERL model receives 0.70 in successful rate on the test unseen set with beam search.

Sensitivity Test

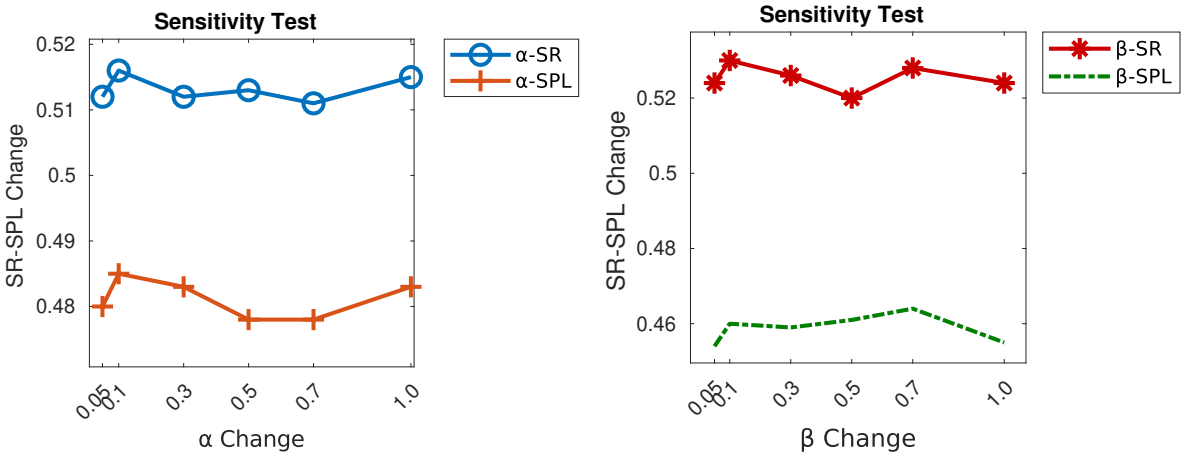

FIGURE 4.4. The sensitivity test of our Soft Expert Reward Learning (SERL) model. The figures show the SR and SPL performance of the model on validation unseen set with different α and β values.

This section presents the performances of SERL model with different α and β weights to trade-off the proposed individual intrinsic reward. Figure 4.4 shows the sensitivity test results, which is evaluated in SR and SPL on validation unseen set. It is clear that SERL generally performs stably w.r.t. the use of different α and β weights. This demonstrates the general stability of our SERL method by setting different hyper-parameters. In general, $\alpha = \beta = 0.1$ is recommended for SERL to achieve effective visual and language navigation performance.

4.5.4 Visualisation

Figure 4.5 shows the actions taken by our baseline agents and proposed SERL agent, respectively. The attention maps over the instruction at each step are also illustrated

FIGURE 4.5. The visualisation of our proposed Soft Expert Reward Learning (SERL) model. The figure shows the comparison between SERL model and the baseline model. The yellow colours in the sentence represents the attention maps over the instruction. The depth of the colours indicates the strength of the attention. The darker the colours, the more attention is put on the specific vocabularies. The check mark means the agents take a same action as the expert; the cross mark represents the opposite.

in the figure. On the left column of the figure, the agent is trained by behaviour cloning solely and it performs correctly at the first three steps. But the agent takes a wrong action at the fourth step and it results in failure navigation in the next three steps. This is because subtle errors will be accumulated at each step by just copy expert demonstrations in the training phase. However, our SERL model can attend over the instruction in a better way and it does not encounter the error accumulation problem in the case.

4.6 Conclusions

In this chapter, we propose a Soft Expert Reward Learning (SERL) model to address the behaviour cloning error accumulation and the reinforcement learning reward engineering issues for VLN task. From the experimental results, we show that our SERL model gains better performance generally than current state-of-the-art methods in both validation unseen and test unseen set on VLN Room-to-Room dataset. The ablation study shows that our proposed the Soft Expert Distillation (SED) module and the Self Perceiving (SP) module are complementary to each other. Moreover, the visualisation experiments further verify the SERL model can overcome the error accumulation problem. In the future, we will further investigate more reward learning methods on VLN task.

Chapter 5

Unsupervised Representation Learning by Predicting Random Distances

5.1 Overview

Deep neural networks have gained tremendous success in a broad range of machine learning tasks due to its remarkable capability to learn semantic-rich features from high-dimensional data. However, they often require large-scale labelled data to successfully learn such features, which significantly hinders their adaptation into unsupervised learning tasks, such as anomaly detection and clustering, and limits their applications into critical domains where obtaining massive labelled data is prohibitively expensive. To enable downstream unsupervised learning on those domains, inspired by the exploration strategies in RL, in this chapter we propose to learn features without using any labelled data by training neural networks to predict data distances in a randomly projected space. Random mapping is a theoretically proven approach to obtain approximately preserved distances. To well predict these random distances, the representation learner is optimised to learn genuine class structures that are implicitly embedded in the randomly projected space. Experimental results on 19 real-world datasets show our learned representations substantially outperform state-of-the-art competing methods in both anomaly detection and clustering tasks.

5.2 Introduction

Unsupervised representation learning aims at automatically extracting expressive feature representations from data without any manually labelled data. Due to the remarkable capability to learn semantic-rich features, deep neural networks have become one widely-used technique to empower a broad range of machine learning tasks. One main issue with these deep learning techniques is that a massive amount of labelled data is typically required to successfully learn these expressive features. As a result, their transformation power is largely reduced for tasks that are unsupervised in nature,

such as anomaly detection and clustering. This is also true to critical domains, such as healthcare and fintech, where collecting massive labelled data is prohibitively expensive and/or is impossible to scale. To bridge this gap, in this work we explore fully unsupervised representation learning techniques to enable downstream unsupervised learning methods on those critical domains.

In recent years, many unsupervised representation learning methods (Mikolov et al., 2013a; Le and Mikolov, 2014; Misra, Zitnick, and Hebert, 2016; Lee et al., 2017; Gidaris, Singh, and Komodakis, 2018) have been introduced, of which most are self-supervised approaches that formulate the problem as an annotation free pretext task. These methods explore easily accessible information, such as temporal or spatial neighbourhood, to design a surrogate supervisory signal to empower the feature learning. These methods have achieved significantly improved feature representations of text/image/video data, but they are often inapplicable to *tabular data* since it does not contain the required temporal or spatial supervisory information. We therefore focus on unsupervised representation learning of high-dimensional tabular data. Although many traditional approaches, such as random projection (Li, Hastie, and Church, 2006), principal component analysis (PCA) (Rahmani and Atia, 2017), manifold learning (Donoho and Grimes, 2003; Hinton and Roweis, 2003) and autoencoder (Vincent et al., 2010), are readily available for handling those data, many of them (Donoho and Grimes, 2003; Hinton and Roweis, 2003; Rahmani and Atia, 2017) are often too computationally costly to scale up to large or high-dimensional data. Approaches like random projection and autoencoder are very efficient but they often fail to capture complex class structures due to its underlying data assumption or weak supervisory signal.

In this chapter, we introduce a Random Distance Prediction (RDP) model which trains neural networks to predict data distances in a randomly projected space. When the distance information captures intrinsic class structure in the data, the representation learner is optimised to learn the class structure to minimise the prediction error. Since distances are concentrated and become meaningless in high dimensional spaces (Beyer et al., 1999), we seek to obtain distances preserved in a projected space to be the supervisory signal. Random mapping is a highly efficient yet theoretical proven approach to obtain such approximately preserved distances. Therefore, we leverage the distances in the randomly projected space to learn the desired features. Intuitively, random mapping preserves rich local proximity information but may also keep misleading proximity when its underlying data distribution assumption is inexact; by minimising the random distance prediction error, RDP essentially leverages the preserved data proximity and the power of neural networks to learn globally consistent proximity and rectify the inconsistent proximity information, resulting in a substantially better representation space than the original space. We show this simple random distance prediction enables us to achieve expressive representations with no manually labelled data. In addition, some task-dependent auxiliary losses can be optionally

added as a complementary supervisory source to the random distance prediction, so as to learn the feature representations that are more tailored for a specific downstream task. In summary, this chapter makes the following three main contributions.

- We propose a random distance prediction formulation, which is very simple yet offers a highly effective supervisory signal for learning expressive feature representations that *optimise* the distance preserving in random projection. The learned features are sufficiently generic and work well in enabling different downstream learning tasks.
- Our formulation is flexible to incorporate task-dependent auxiliary losses that are complementary to random distance prediction to further enhance the learned features, i.e., features that are specifically optimised for a downstream task while at the same time preserving the generic proximity as much as possible.
- As a result, we show that our instantiated model termed RDP enables substantially better performance than state-of-the-art competing methods in two key unsupervised tasks, anomaly detection and clustering, on 19 real-world high-dimensional tabular datasets.

5.3 Background

Self-supervised Learning. Self-supervised learning has been recently emerging as one of the most popular and effective approaches for representation learning. Many of the self-supervised methods learn high-level representations by predicting some sort of 'context' information, such as spatial or temporal neighbourhood information. For example, the popular distributed representation learning techniques in NLP, such as CBOW/skip-gram (Mikolov et al., 2013a) and phrase/sentence embeddings in (Mikolov et al., 2013b; Le and Mikolov, 2014; Hill, Cho, and Korhonen, 2016), learn the representations by predicting the text pieces (e.g., words/phrases/sentences) using its surrounding pieces as the context. In image processing, the pretext task can be the prediction of a patch of missing pixels (Pathak et al., 2016; Zhang, Isola, and Efros, 2017) or the relative position of two patches (Doersch, Gupta, and Efros, 2015). Also, a number of studies (Goroshin et al., 2015; Misra, Zitnick, and Hebert, 2016; Lee et al., 2017; Oord, Li, and Vinyals, 2018) explore temporal contexts to learn representations from video data, e.g., by learning the temporal order of sequential frames. Some other methods (Agrawal, Carreira, and Malik, 2015; Zhou et al., 2017; Gidaris, Singh, and Komodakis, 2018) are built upon a discriminative framework which aims at discriminating the images before and after some transformation, e.g., ego motion in video data (Agrawal, Carreira, and Malik, 2015; Zhou et al., 2017) and rotation of images (Gidaris, Singh, and Komodakis, 2018). There have also been popular to use generative adversarial networks (GANs) to learn features (Radford, Metz, and Chintala, 2015; Chen et al., 2016). The above methods have demonstrated powerful

capability to learn semantic representations. However, most of them use the supervisory signals available in image/video data only, which limits their application into other types of data, such as traditional tabular data. Although our method may also work on image/video data, we focus on handling high-dimensional tabular data to bridge this gap.

Other Approaches. There have been several well-established unsupervised representation learning approaches for handling tabular data, such as random projection (Arriaga and Vempala, 1999; Bingham and Mannila, 2001; Li, Hastie, and Church, 2006), PCA (Wold, Esbensen, and Geladi, 1987; Schölkopf, Smola, and Müller, 1997; Rahmani and Atia, 2017), manifold learning (Roweis and Saul, 2000; Donoho and Grimes, 2003; Hinton and Roweis, 2003; McInnes, Healy, and Melville, 2018) and autoencoder (Hinton and Salakhutdinov, 2006; Vincent et al., 2010). One notorious issue of PCA or manifold learning approaches is their prohibitive computational cost in dealing with large-scale high-dimensional data due to the costly neighbourhood search and/or eigen decomposition. Random projection is a computationally efficient approach, supported by proven distance preservation theories such as the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984). We show that the preserved distances by random projection can be harvested to effectively supervise the representation learning. Autoencoder networks are another widely-used efficient feature learning approach which learns low-dimensional representations by minimising reconstruction errors. One main issue with autoencoders is that they focus on preserving global information only, which may result in loss of local structure information. Some representation learning methods are specifically designed for anomaly detection (Pang et al., 2018; Zong et al., 2018; Burda et al., 2019). By contrast, we aim at generic representations learning while being flexible to incorporate optionally task-dependent losses to learn task-specific semantic-rich representations.

5.4 Random Distance Prediction Model

5.4.1 The Proposed Formulation and The Instantiated Model

We propose to learn representations by training neural networks to predict distances in a randomly projected space without manually labelled data. The key intuition is that, given some distance information that faithfully encapsulates the underlying class structure in the data, the representation learner is forced to learn the class structure in order to yield distances that are as close as the given distances. Our proposed framework is illustrated in Figure 5.1. Specifically, given data points $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^D$, we first feed them into a weight-shared Siamese-style neural network $\phi(\mathbf{x}; \Theta)$. $\phi : \mathbb{R}^D \mapsto \mathbb{R}^M$ is a representation learner with the parameters Θ to map the data onto a M-dimensional new space. Then we formulate the subsequent step as a distance

prediction task and define a loss function as:

$$L_{rdp}(\mathbf{x}_i, \mathbf{x}_j) = l(\langle \phi(\mathbf{x}_i; \Theta), \phi(\mathbf{x}_j; \Theta) \rangle, \langle \eta(\mathbf{x}_i), \eta(\mathbf{x}_j) \rangle), \tag{5.1}$$

where η is an existing projection method and l is a function of the difference between its two inputs.

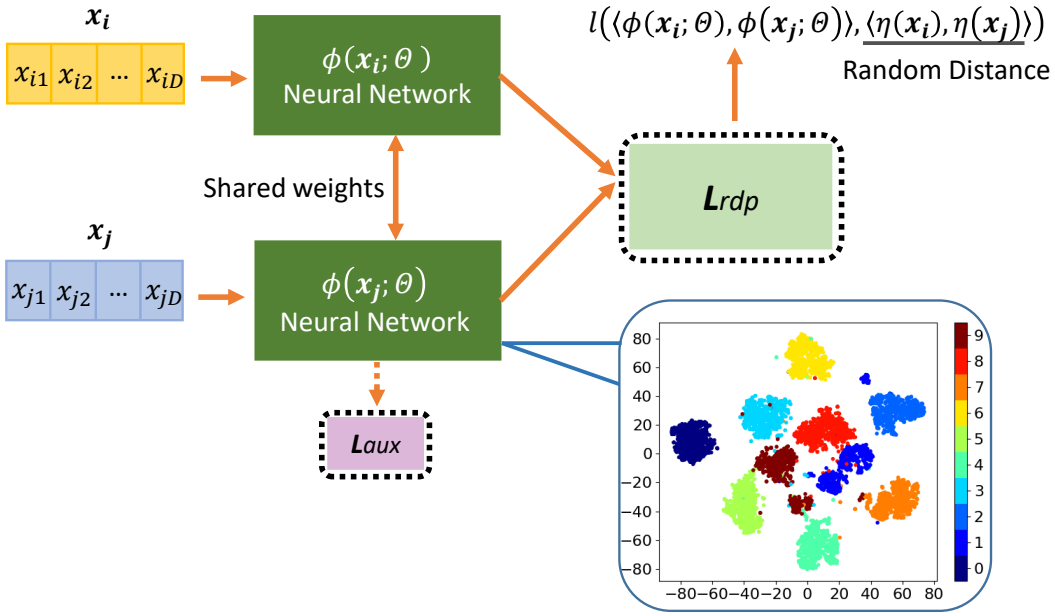

FIGURE 5.1. The proposed random distance prediction (RDP) framework. Specifically, a weight-shared two-branch neural network ϕ first projects $\mathbf{x}_i$ and $\mathbf{x}_j$ onto a new space, in which we aim to minimise the random distance prediction loss L_{rdp}, i.e., the difference between the learned distance $\langle \phi(\mathbf{x}_i; \Theta), \phi(\mathbf{x}_j; \Theta) \rangle$ and a predefined distance $\langle \eta(\mathbf{x}_i), \eta(\mathbf{x}_j) \rangle$ (η denotes an existing random mapping). L_{aux} is an auxiliary loss that is optionally applied to one network branch to learn complementary information w.r.t. L_{rdp}. The lower right figure presents a 2-D t-SNE (Hinton and Roweis, 2003) visualisation of the features learned by RDP on a small toy dataset *optdigits* with 10 classes.

Here one key ingredient is how to obtain trustworthy distances via η. Also, to efficiently optimise the model, the distance derivation needs to be computationally efficient. In this work, we use the inner products in a randomly projected space as the source of distance/similarity since it is very efficient and there is strong theoretical support of its capacity in preserving the genuine distance information. Thus, our instantiated model RDP specifies $L_{rdp}(\mathbf{x}_i, \mathbf{x}_j)$ as follows[1]:

$$L_{rdp}(\mathbf{x}_i, \mathbf{x}_j) = \left(\phi(\mathbf{x}_i; \Theta) \cdot \phi(\mathbf{x}_j; \Theta) - \eta(\mathbf{x}_i) \cdot \eta(\mathbf{x}_j)\right)^2, \tag{5.2}$$

where ϕ is implemented by multilayer perceptron for dealing with tabular data and $\eta : \mathbb{R}^D \mapsto \mathbb{R}^K$ is an off-the-shelf random data mapping function (see Sections 5.5.1 and 5.5.2 for detail). Despite its simplicity, this loss offers a powerful supervisory signal to

[1]Since we operate on real-valued vector space, the inner product is implemented by the dot product. The dot product is used hereafter to simplify the notation.

learn semantic-rich feature representations that substantially optimise the underlying distance preserving in η (see Section 5.5.3 for detail).

5.4.2 Flexibility to Incorporate Task-dependent Complementary Auxiliary Loss

Minimising L_{rdp} learns to preserve pairwise distances that are critical to different learning tasks. Moreover, our formulation is flexible to incorporate a task-dependent auxiliary loss L_{aux}, such as reconstruction loss (Hinton and Salakhutdinov, 2006) for clustering or novelty loss (Burda et al., 2019) for anomaly detection, to complement the proximity information and enhance the feature learning.

For clustering, an auxiliary reconstruction loss is defined as:

$$L_{aux}^{clu}(\mathbf{x}) = \left(\mathbf{x} - \phi'(\phi(\mathbf{x};\Theta);\Theta')\right)^2, \tag{5.3}$$

where ϕ is an encoder and $\phi' : \mathbb{R}^M \mapsto \mathbb{R}^D$ is a decoder. This loss may be optionally added into RDP to better capture global feature representations.

Similarly, in anomaly detection a novelty loss may be optionally added, which is defined as:

$$L_{aux}^{ad}(\mathbf{x}) = \left(\phi(\mathbf{x};\Theta) - \eta(\mathbf{x})\right)^2. \tag{5.4}$$

By using a fixed η, minimising L_{aux}^{ad} helps learn the frequency of underlying patterns in the data (Burda et al., 2019), which is an important complementary supervisory source for the sake of anomaly detection. As a result, anomalies or novel points are expected to have substantially larger $\left(\phi(\mathbf{x};\Theta^\star) - \eta(\mathbf{x})\right)^2$ than normal points, so this value can be directly leveraged to detect anomalies.

Note since L_{aux}^{ad} involves a mean squared error between two vectors, the dimension of the projected space resulted by ϕ and η is required to be equal in this case. Therefore, when this loss is added into RDP, the M in ϕ and K in η need to be the same. We do not have this constraint in other cases.

5.5 Theoretical Analysis of RDP

This section shows the proximity information can be well approximated using inner products in two types of random projection spaces. This is a key theoretical foundation to RDP. Also, to accurately predict these distances, RDP is forced to learn the genuine class structure in the data.

5.5.1 When Linear Projection Is Used

Random projection is a simple yet very effective linear feature mapping technique which has proven the capability of distance preservation. Let $\mathcal{X} \subset \mathbb{R}^{N\times D}$ be a set of N data points, random projection uses a random matrix $\mathbf{A} \subset \mathbb{R}^{K\times D}$ to project the data

onto a lower K-dimensional space by $\mathcal{X}' = \mathbf{A}\mathcal{X}^\intercal$. The Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984) guarantees the data points can be mapped to a randomly selected space of suitably lower dimension with the distances between the points are approximately preserved. More specifically, let $\epsilon \in (0, \frac{1}{2})$ and $K = \frac{20 \log n}{\epsilon^2}$. There exists a linear mapping $f : \mathbb{R}^D \mapsto \mathbb{R}^K$ such that for all $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$:

$$(1-\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2 \leq (1+\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2. \tag{5.5}$$

Furthermore, assume the entries of the matrix $\mathbf{A}$ are sampled independently from a Gaussian distribution $\mathcal{N}(0,1)$. Then, the norm of $\mathbf{x} \in \mathbb{R}^D$ can be preserved as:

$$\Pr\left((1-\epsilon)||\mathbf{x}||^2 \leq ||\frac{1}{\sqrt{K}}\mathbf{A}\mathbf{x}||^2 \leq (1+\epsilon)||\mathbf{x}||^2\right) \geq 1 - 2e^{\frac{-(\epsilon^2-\epsilon^3)K}{4}}. \tag{5.6}$$

Under such random projections, the norm preservation helps well preserve the inner products:

$$\Pr\left(|\hat{\mathbf{x}}_i \cdot \hat{\mathbf{x}}_j - f(\hat{\mathbf{x}}_i) \cdot f(\hat{\mathbf{x}}_j)| \geq \epsilon\right) \leq 4e^{\frac{-(\epsilon^2-\epsilon^3)K}{4}}, \tag{5.7}$$

where $\hat{\mathbf{x}}$ is a normalised $\mathbf{x}$ such that $||\hat{\mathbf{x}}|| \leq 1$.

The proofs of Eqns. (5.5), (5.6) and (5.7) can be found in (Vempala, 1998).

Eqn. (5.7) states that the inner products in the randomly projected space can largely preserve the inner products in the original space, particularly when the projected dimension K is large.

5.5.2 When Non-linear Projection Is Used

Here we show that some non-linear random mapping methods are approximate to kernel functions which are a well-established approach to obtain reliable distance/similarity information. The key to this approach is the kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, which is defined as $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle$, where ψ is a feature mapping function but needs not to be explicitly defined and $\langle \cdot, \cdot \rangle$ denotes a suitable inner product. A non-linear kernel function such as polynomial or radial basis function (RBF) kernel is typically used to project linear-inseparable data onto a linear-separable space.

The relation between non-linear random mapping and kernel methods is justified in (Rahimi and Recht, 2008), which shows that an explicit randomised mapping function $g : \mathbb{R}^D \mapsto \mathbb{R}^K$ can be defined to project the data points onto a low-dimensional Euclidean inner product space such that the inner products in the projected space approximate the kernel evaluation:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j) \rangle \approx g(\mathbf{x}_i) \cdot g(\mathbf{x}_j). \tag{5.8}$$

Let $\mathbf{A}$ be the mapping matrix. Then to achieve the above approximation, $\mathbf{A}$ is required to be drawn from Fourier transform and shift-invariant functions such as cosine function are finally applied to $\mathbf{Ax}$ to yield a real-valued output. By transforming the two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in this manner, their inner product $g(\mathbf{x}_i) \cdot g(\mathbf{x}_j)$ is an unbiased estimator of $k(\mathbf{x}_i, \mathbf{x}_j)$.

5.5.3 Learning Class Structure By Random Distance Prediction

Our model using only the random distances as the supervisory signal can be formulated as:

$$\arg\min_{\Theta} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}} \left(\phi(\mathbf{x}_i; \Theta) \cdot \phi(\mathbf{x}_j; \Theta) - y_{ij}\right)^2, \tag{5.9}$$

where $y_{ij} = \eta(\mathbf{x}_i) \cdot \eta(\mathbf{x}_j)$. Let $\mathbf{Y}_\eta \in \mathbb{R}^{N \times N}$ be the distance/similarity matrix of the N data points resulted by η. Then to minimise the prediction error in Eqn. (5.9), ϕ is optimised to learn the underlying class structure embedded in $\mathbf{Y}$. As shown in the properties in Eqns. (5.7) and (5.8), $\mathbf{Y}_\eta$ can effectively preserve local proximity information when η is set to be either the random projection-based f function or the kernel method-based g function. However, those proven η is often built upon some underlying data distribution assumption, e.g., Gaussian distribution in random projection or Gaussian RBF kernel, so the η-projected features can preserve misleading proximity when the distribution assumption is inexact. In this case, $\mathbf{Y}_\eta$ is equivalent to the imperfect ground truth with partial noise. Then optimisation with Eqn. (5.9) is to leverage the power of neural networks to learn consistent local proximity information and rectify inconsistent proximity, resulting in a significantly optimised distance preserving space. The resulting space conveys substantially richer semantics than the η projected space when $\mathbf{Y}_\eta$ contains sufficient genuine supervision information.

5.6 Experiments

This section evaluates the learned representations through two typical unsupervised tasks: anomaly detection and clustering. Some preliminary results of classification can be found in Appendix 5.15.

5.6.1 Performance Evaluation in Anomaly Detection

Experimental Settings

Our RDP model is compared with five state-of-the-art methods, including iForest (Liu, Ting, and Zhou, 2008), autoencoder (AE) (Hinton and Salakhutdinov, 2006), REPEN (Pang et al., 2018), DAGMM (Zong et al., 2018) and RND (Burda et al., 2019). iForest and AE are two of the most popular baselines. The other three methods learn representations specifically for anomaly detection.

As shown in Table 5.1, the comparison is performed on 14 publicly available datasets of various domains, including network intrusion, credit card fraud detection, disease detection and bank campaigning. Many of the datasets contain real anomalies, including *DDoS*, *Donors*, *Backdoor*, *Creditcard*, *Lung*, *Probe* and *U2R*. Following (Liu, Ting, and Zhou, 2008; Pang et al., 2018; Zong et al., 2018), the rare class(es) is treated as anomalies in the other datasets to create semantically real anomalies. The Area Under Receiver Operating Characteristic Curve (AUC-ROC) and the Area Under Precision-Recall Curve (AUC-PR) are used as our performance metrics. Larger AUC-ROC/AUC-PR indicates better performance. The reported performance is averaged over 10 independent runs.

TABLE 5.1. AUC-ROC (mean±std) performance of RDP and its five competing methods on 14 datasets.

Data Characteristics				Our Method RDP and Its Five Competing Methods					
Data	**N**	**D**	**Anomaly (%)**	**iForest**	**AE**	**REPEN**	**DAGMM**	**RND**	**RDP**
DDoS	464,976	66	3.75%	0.880 ± 0.018	0.901 ± 0.000	0.933 ± 0.002	0.766 ± 0.019	0.852 ± 0.011	**0.942 ± 0.008**
Donors	619,326	10	5.92%	0.774 ± 0.010	0.812 ± 0.011	0.777 ± 0.075	0.763 ± 0.110	0.847 ± 0.011	**0.962 ± 0.011**
Backdoor	95,329	196	2.44%	0.723 ± 0.029	0.806 ± 0.007	0.857 ± 0.001	0.813 ± 0.035	**0.935 ± 0.002**	0.910 ± 0.021
Ad	3,279	1,555	13.99%	0.687 ± 0.021	0.703 ± 0.000	0.853 ± 0.001	0.500 ± 0.000	0.812 ± 0.002	**0.887 ± 0.003**
Apascal	12,695	64	1.38%	0.514 ± 0.051	0.623 ± 0.005	0.813 ± 0.004	0.710 ± 0.020	0.685 ± 0.019	**0.823 ± 0.007**
Bank	41,188	62	11.26%	0.713 ± 0.021	0.666 ± 0.000	0.681 ± 0.001	0.616 ± 0.014	0.690 ± 0.006	**0.758 ± 0.007**
Celeba	202,599	39	2.24%	0.693 ± 0.014	0.735 ± 0.002	0.802 ± 0.002	0.680 ± 0.067	0.682 ± 0.029	**0.860 ± 0.006**
Census	299,285	500	6.20%	0.599 ± 0.019	0.602 ± 0.000	0.542 ± 0.003	0.502 ± 0.003	**0.661 ± 0.003**	0.653 ± 0.004
Creditcard	284,807	29	0.17%	0.948 ± 0.005	0.948 ± 0.000	0.950 ± 0.001	0.877 ± 0.005	0.945 ± 0.001	**0.957 ± 0.005**
Lung	145	3,312	4.13%	0.893 ± 0.057	0.953 ± 0.004	0.949 ± 0.002	0.830 ± 0.087	0.867 ± 0.031	**0.982 ± 0.006**
Probe	64,759	34	6.43%	0.995 ± 0.001	0.997 ± 0.000	0.997 ± 0.000	0.953 ± 0.008	0.975 ± 0.000	**0.997 ± 0.000**
R8	3,974	9,467	1.28%	0.841 ± 0.023	0.835 ± 0.000	**0.910 ± 0.000**	0.760 ± 0.066	0.883 ± 0.006	0.902 ± 0.002
Secom	1,567	590	6.63%	0.548 ± 0.019	0.526 ± 0.000	0.510 ± 0.004	0.513 ± 0.010	0.541 ± 0.006	**0.570 ± 0.004**
U2R	60,821	34	0.37%	**0.988 ± 0.001**	0.987 ± 0.000	0.978 ± 0.000	0.945 ± 0.028	0.981 ± 0.001	0.986 ± 0.001

Our RDP model uses the optional novelty loss for anomaly detection task by default. Similar to RND, given a data point $\mathbf{x}$, its anomaly score in RDP is defined as the mean squared error between the two projections resulted by $\phi(\mathbf{x}; \Theta^\star)$ and $\eta(\mathbf{x})$. Also, a boosting process is used to filter out 5% likely anomalies per iteration to iteratively improve the modelling of RDP. This is because the modelling is otherwise largely biased when anomalies are presented. In the ablation study in Section 5.6.1, we will show the contribution of all these components.

Comparison to the state-of-the-art competing methods

The AUC-ROC and AUC-PR results are respectively shown in Tables 1 and 2. RDP outperforms all the five competing methods in both of AUC-ROC and AUC-PR in at least 12 out of 14 datasets. This improvement is statistically significant at the 95% confidence level according to the two-tailed sign test (Demšar, 2006). Remarkably, RDP obtains more than 10% AUC-ROC/AUC-PR improvement over the best competing method on six datasets, including *Donors*, *Ad*, *Bank*, *Celeba*, *Lung* and *U2R*. RDP can be thought as a high-level synthesis of REPEN and RND, because REPEN leverages a pairwise distance-based ranking loss to learn representations for anomaly detection while RND is built using L^{ad}_{aux}. In nearly all the datasets, RDP well leverages both L_{rdp} and L^{ad}_{aux} to achieve significant improvement over both REPEN and RND.

In very limited cases, such as on datasets *Backdoor* and *Census* where RND performs very well while REPEN performs less effectively, RDP is slightly downgraded due to the use of L_{rdp}. In the opposite case, such as *Probe*, on which REPEN performs much better than RND, the use of L^{ad}_{aux} may drag down the performance of RDP a bit.

TABLE 5.2. AUC-PR (mean±std) performance of RDP and its five competing methods on 14 datasets.

Data	iForest	AE	REPEN	DAGMM	RND	RDP
DDoS	0.141 ± 0.020	0.248 ± 0.001	0.300 ± 0.012	0.038 ± 0.000	0.110 ± 0.015	**0.301 ± 0.028**
Donors	0.124 ± 0.006	0.138 ± 0.007	0.120 ± 0.032	0.070 ± 0.024	0.201 ± 0.033	**0.432 ± 0.061**
Backdoor	0.045 ± 0.007	0.065 ± 0.004	0.129 ± 0.001	0.034 ± 0.023	**0.433 ± 0.015**	0.305 ± 0.008
Ad	0.363 ± 0.061	0.479 ± 0.000	0.600 ± 0.002	0.140 ± 0.000	0.473 ± 0.009	**0.726 ± 0.007**
Apascal	0.015 ± 0.002	0.023 ± 0.001	0.041 ± 0.001	0.023 ± 0.009	0.021 ± 0.005	**0.042 ± 0.003**
Bank	0.293 ± 0.023	0.264 ± 0.001	0.276 ± 0.001	0.150 ± 0.020	0.258 ± 0.006	**0.364 ± 0.013**
Celeba	0.060 ± 0.006	0.082 ± 0.001	0.081 ± 0.001	0.037 ± 0.017	0.068 ± 0.010	**0.104 ± 0.006**
Census	0.071 ± 0.004	0.072 ± 0.000	0.064 ± 0.005	0.061 ± 0.001	0.081 ± 0.001	**0.086 ± 0.001**
Creditcard	0.145 ± 0.031	**0.382 ± 0.004**	0.359 ± 0.014	0.010 ± 0.012	0.290 ± 0.012	0.363 ± 0.011
Lung	0.379 ± 0.092	0.565 ± 0.022	0.429 ± 0.005	0.042 ± 0.003	0.381 ± 0.104	**0.705 ± 0.028**
Probe	0.923 ± 0.011	0.964 ± 0.002	**0.964 ± 0.000**	0.409 ± 0.153	0.609 ± 0.014	0.955 ± 0.002
R8	0.076 ± 0.018	0.097 ± 0.006	0.083 ± 0.000	0.019 ± 0.011	0.134 ± 0.031	**0.146 ± 0.017**
Secom	0.106 ± 0.007	0.093 ± 0.000	0.091 ± 0.001	0.066 ± 0.002	0.086 ± 0.002	**0.096 ± 0.001**
U2R	0.180 ± 0.018	0.230 ± 0.004	0.116 ± 0.007	0.025 ± 0.019	0.217 ± 0.011	**0.261 ± 0.005**

Ablation Study

This section examines the contribution of L_{rdp}, L^{ad}_{aux} and the boosting process to the performance of RDP. The experimental results in AUC-ROC are given in Table 5.3, where RDP\X means the RDP variant that removes the 'X' module from RDP. In the last two columns, *Org_SS* indicates that we directly use the distance information calculated in the original space as the supervisory signal, while *SRP_SS* indicates that we use SRP to obtain the distances as the supervisory signal. It is clear that the full RDP model is the best performer. Using the L_{rdp} loss only, i.e., RDP$\backslash L^{ad}_{aux}$, can achieve performance substantially better than, or comparably well to, the five competing methods in Table 5.1. This is mainly because the L_{rdp} loss alone can effectively force our representation learner to learn the underlying class structure on most datasets so as to minimise its prediction error. The use of L^{ad}_{aux} and boosting process well complement the L_{rdp} loss on the other datasets.

In terms of supervisory source, RDP and SRP_SS perform substantially better than Org_SS on most datasets. This is because the distances in both the non-linear random projection in RDP and the linear projection in SRP_SS well preserve the distance information, enabling RDP to effectively learn much more faithful class structure than that working on the original space.

5.6.2 Performance Evaluation in Clustering

Experimental Settings

For clustering, RDP is compared with four state-of-the-art unsupervised representation learning methods in four different areas, including HLLE (Donoho and Grimes, 2003) in manifold learning, Sparse Random Projection (SRP) (Li, Hastie, and Church,

TABLE 5.3. AUC-ROC results of anomaly detection (see Appendix 5.10 for similar AUC-PR results).

	Decomposition				Supervision Signal	
Data	**RDP**	**RDP\L_{rdp}**	**RDP\L^{ad}_{aux}**	**RDP\Boosting**	**Org_SS**	**SRP_SS**
DDoS	**0.942 ± 0.008**	0.852 ± 0.011	0.931 ± 0.003	0.866 ± 0.011	0.924 ± 0.006	0.927 ± 0.005
Donors	**0.962 ± 0.011**	0.847 ± 0.011	0.737 ± 0.006	0.910 ± 0.013	0.728 ± 0.005	0.762 ± 0.016
Backdoor	0.910 ± 0.021	0.935 ± 0.002	0.872 ± 0.012	**0.943 ± 0.002**	0.875 ± 0.002	0.882 ± 0.010
Ad	**0.887 ± 0.003**	0.812 ± 0.002	0.718 ± 0.005	0.818 ± 0.002	0.696 ± 0.003	0.740 ± 0.008
Apascal	**0.823 ± 0.007**	0.685 ± 0.019	0.732 ± 0.007	0.804 ± 0.021	0.604 ± 0.032	0.760 ± 0.030
Bank	**0.758 ± 0.007**	0.690 ± 0.006	0.684 ± 0.004	0.736 ± 0.009	0.684 ± 0.002	0.688 ± 0.015
Celeba	**0.860 ± 0.006**	0.682 ± 0.029	0.709 ± 0.005	0.794 ± 0.017	0.667 ± 0.033	0.734 ± 0.027
Census	0.653 ± 0.004	**0.661 ± 0.003**	0.626 ± 0.006	0.661 ± 0.001	0.636 ± 0.006	0.560 ± 0.006
Creditcard	**0.957 ± 0.005**	0.945 ± 0.001	0.950 ± 0.000	0.956 ± 0.003	0.947 ± 0.001	0.949 ± 0.003
Lung	**0.982 ± 0.006**	0.867 ± 0.031	0.911 ± 0.006	0.968 ± 0.018	0.884 ± 0.018	0.928 ± 0.008
Probe	0.997 ± 0.000	0.975 ± 0.000	**0.998 ± 0.000**	0.978 ± 0.001	0.995 ± 0.000	0.997 ± 0.001
R8	0.902 ± 0.002	0.883 ± 0.006	0.867 ± 0.003	0.895 ± 0.004	0.830 ± 0.005	**0.904 ± 0.005**
Secom	**0.57 ± 0.004**	0.541 ± 0.006	0.544 ± 0.011	0.563 ± 0.008	0.512 ± 0.007	0.530 ± 0.016
U2R	0.986 ± 0.001	0.981 ± 0.001	0.987 ± 0.000	**0.988 ± 0.002**	0.987 ± 0.000	0.981 ± 0.002
#wins/draws/losses (RDP vs.)		13/0/1	13/0/1	12/0/2	10/2/2	6/0/8

2006) in random projection, autoencoder (AE) (Hinton and Salakhutdinov, 2006) in data reconstruction-based neural network methods and Coherence Pursuit (COP) (Rahmani and Atia, 2017) in robust PCA. These representation learning methods are first used to yield the new representations, and K-means (Hartigan and Wong, 1979) is then applied to the representations to perform clustering. Two widely-used clustering performance metrics, Normalised Mutual Info (NMI) score and F-score, are used. Larger NMI or F-score indicates better performance. The clustering performance in the original feature space, denoted as Org, is used as a baseline. As shown in Table 5.4, five high-dimensional real-world datasets are used. Some of the datasets are image/text data. Since here we focus on the performance on tabular data, they are converted into tabular data using simple methods, i.e., by treating each pixel as a feature unit for image data or using bag-of-words representation for text data[2]. The reported NMI score and F-score are averaged over 30 times to address the randomisation issue in K-means clustering. In this section RDP adds the reconstruction loss L^{clu}_{aux} by default, but RDP also works very well without the use of L^{clu}_{aux}.

Comparison to the-state-of-the-art competing methods

Table 5.4 shows the NMI and F-score performance of K-means clustering. Our method RDP enables K-means to achieve the best performance on three datasets and ranks second in the other two datasets. RDP-enabled clustering performs substantially and consistently better than that based on AE in terms of both NMI and F-score. This demonstrates that the random distance loss enables RDP to effectively capture some class structure in the data which cannot be captured by using the reconstruction loss. RDP also consistently outperforms the random projection method, SRP, and the robust PCA method, COP. It is interesting that K-means clustering performs best in the original space on *Sector*. This may be due to that this data contains many relevant features, resulting in no obvious curse of dimensionality issue. *Olivetti* may

[2]RDP can also build upon advanced representation learning methods for the data transformation, for which some interesting preliminary results are presented in Appendix 5.14.

contain complex manifolds which require extensive neighbourhood information to find them, so only HLLE can achieve this goal in such cases. Nevertheless, RDP performs much more stably than HLLE across the five datasets.

TABLE 5.4. NMI and F-score performance of K-means on the original space and projected spaces.

Data Characteristics			**NMI Performance**					
Data	**N**	**D**	**Org**	**HLLE**	**SRP**	**AE**	**COP**	**RDP**
R8	7,674	17,387	0.524 ± 0.047	0.004 ± 0.001	0.459 ± 0.031	0.471 ± 0.043	0.025 ± 0.003	**0.539 ± 0.040**
20news	18,846	130,107	0.080 ± 0.004	0.017 ± 0.000	0.075 ± 0.002	0.075 ± 0.006	0.027 ± 0.040	**0.084 ± 0.005**
Olivetti	400	4,096	0.778 ± 0.014	**0.841 ± 0.011**	0.774 ± 0.011	0.782 ± 0.010	0.333 ± 0.018	0.805 ± 0.012
Sector	9,619	55,197	**0.336 ± 0.008**	0.122 ± 0.004	0.273 ± 0.011	0.253 ± 0.010	0.129 ± 0.014	0.305 ± 0.007
RCV1	20,242	47,236	0.154 ± 0.000	0.006 ± 0.000	0.134 ± 0.024	0.146 ± 0.010	N/A	**0.165 ± 0.000**
Data Characteristics			**F-score Performance**					
Data	**N**	**D**	**Org**	**HLLE**	**SRP**	**AE**	**COP**	**RDP**
R8	7,674	17,387	0.185 ± 0.189	0.085 ± 0.000	0.317 ± 0.045	0.312 ± 0.068	0.088 ± 0.002	**0.360 ± 0.055**
20news	18,846	130,107	0.116 ± 0.006	0.007 ± 0.000	0.109 ± 0.006	0.083 ± 0.010	0.009 ± 0.004	**0.119 ± 0.006**
Olivetti	400	4,096	0.590 ± 0.029	**0.684 ± 0.024**	0.579 ± 0.022	0.602 ± 0.023	0.117 ± 0.011	0.638 ± 0.026
Sector	9,619	55,197	**0.208 ± 0.008**	0.062 ± 0.001	0.187 ± 0.009	0.184 ± 0.010	0.041 ± 0.004	0.191 ± 0.007
RCV1	20,242	47,236	0.519 ± 0.000	0.342 ± 0.000	0.508 ± 0.003	0.514 ± 0.057	N/A	**0.572 ± 0.003**

TABLE 5.5. F-score performance of K-means clustering (see similar NMI results in Appendix 5.11).

	Decomposition			**Supervision Signal**	
Data	**RDP**	**RDP**$\backslash L_{rdp}$	**RDP**$\backslash L_{aux}^{clu}$	**Org_SS**	**SRP_SS**
R8	0.360 ± 0.055	0.312 ± 0.068	0.330 ± 0.052	0.359 ± 0.028	**0.363 ± 0.046**
20news	**0.119 ± 0.006**	0.083 ± 0.010	0.117 ± 0.005	0.111 ± 0.005	0.111 ± 0.007
Olivetti	**0.638 ± 0.026**	0.602 ± 0.023	0.597 ± 0.019	0.610 ± 0.022	0.601 ± 0.023
Sector	0.191 ± 0.007	0.184 ± 0.010	**0.217 ± 0.007**	0.181 ± 0.007	0.186 ± 0.009
RCV1	**0.572 ± 0.003**	0.514 ± 0.057	0.526 ± 0.011	0.523 ± 0.003	0.532 ± 0.001

Ablation Study

Similar to anomaly detection, this section examines the contribution of the two loss functions L_{rdp} and L_{aux}^{clu} to the performance of RDP, as well as the impact of different supervisory sources on the performance. The F-score results of this experiment are shown in Table 5.5, in which the notations have exactly the same meaning as in Table 5.3. The full RDP model that uses both L_{rdp} and L_{aux}^{clu} performs more favourably than its two variants, RDP$\backslash L_{rdp}$ and RDP$\backslash L_{aux}^{clu}$, but it is clear that using L_{rdp} only performs very comparably to the full RDP. However, using L_{aux}^{clu} only may result in large performance drops in some datasets, such as *R8*, *20news* and *Olivetti*. This indicates L_{rdp} is a more important loss function to the overall performance of the full RDP model. In terms of supervisory source, distances obtained by the non-linear random projection in RDP are much more effective than the two other sources on some datasets such as *Olivetti* and *RCV1*. Three different supervisory sources are very comparable on the other three datasets.

5.7 Conclusion

We introduce a novel Random Distance Prediction (RDP) model which learns features in a fully unsupervised fashion by predicting data distances in a randomly projected space. The key insight is that random mapping is a theoretical proven approach to obtain approximately preserved distances, and to well predict these random distances, the representation learner is optimised to learn consistent preserved proximity information while at the same time rectifying inconsistent proximity, resulting in representations with optimised distance preserving. Our idea is justified by thorough experiments in two unsupervised tasks, anomaly detection and clustering, which show RDP-enabled anomaly detectors and clustering substantially outperform their counterparts on 19 real-world datasets. We plan to extend RDP to other types of data to broaden its application scenarios.

5.8 Implementation Details

RDP-enabled Anomaly Detection. The RDP consists of one fully connected layer with 50 hidden units, followed by a leaky-ReLU layer. It is trained using Stochastic Gradient Descent (SGD) as its optimiser for 200 epochs, with 192 samples per batch. The learning rate is fixed to 0.1. We repeated the boosting process 30 times to obtain statistically stable results. In order to have fair comparisons, we also adapt the competing methods AE, REPEN, DAGMM and RND into ensemble methods and perform the experiments using an ensemble size of 30.

RDP-enabled Clustering. RDP uses a similar network architecture and optimisation settings as the one used in anomaly detection, i.e., the network consists of one fully connected layer, followed by a leaky-ReLU layer, which is optimised by SGD with 192 samples per batch and 0.1 learning rate. Compared to anomaly detection, more semantic information is required for clustering algorithms to work well, so the network consists of 1,024 hidden units and is trained for 1,000 epochs. Clustering is a significant yet common analysis method, which aims at grouping samples close to each other into the same clusters and separating far away data points into different clusters. Compared to anomaly detection that often requires pattern frequency information, clustering has a higher requirement of the representation expressiveness. Therefore, if the representative ability of a model is strong enough, it should also be able to learn representations that enable clustering to work well on the projected space.

Note that the representation dimension M in the ϕ function and the projection dimension K in the η function are set to be the same to alleviate parameter tuning. This means that $M = K = 50$ is used in anomaly detection and $M = K = 1024$ is used in clustering. We have also tried deeper network structures, but they worked less effectively than the shallow networks in both anomaly detection and clustering.

This may be because the supervisory signal is not strong enough to effectively learn deeper representations. We show in Appendix 5.12 that RDP performs stably w.r.t. a range of representation dimensions in both anomaly detection and clustering tasks.

The runtime of RDP at the testing stage is provided in Appendix 5.13 with that of the competing methods as baselines. For both anomaly detection and clustering tasks, RDP achieves very comparable time complexity to the most efficient competing methods (see Tables 5.10 and 5.11 in Appendix 5.13 for detail).

5.9 Datasets

The statistics and the accessible links of the datasets used in the anomaly detection and clustering tasks are respectively presented in Tables 5.6 and 5.7. *DDoS* is a dataset containing DDoS attacks and normal network flows. *Donors* is from KDD Cup 2014, which is used for detecting a very small number of outstanding donors projects. *Backdoor* contains backdoor network attacks derived from the UNSW-NB15 dataset. *Creditcard* is a credit card fraud detection dataset. *Lung* contains data records of lung cancer patients and normal patients. *Probe* and *U2R* are derived from KDD Cup 99, in which probing and user-to-root attacks are respectively used as anomalies against the normal network flows. The above datasets contain real anomalies. Following (Liu, Ting, and Zhou, 2008; Pang et al., 2018; Zong et al., 2018), the other anomaly detection datasets are transformed from classification datasets by using the rare class(es) as the anomaly class, which generates semantically real anomalies.

Table 5.6. Datasets used in the anomaly detection task

Data	N	D	Anomaly (%)	Link
DDoS	464,976	66	3.75%	http://www.csmining.org/cdmc2018/index.php
Donors	619,326	10	5.92%	https://www.kaggle.com/c/kdd-cup-2014-predicting-excitement-at-donors-choose
Backdoor	95,329	196	2.44%	https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity
Ad	3,279	1,555	13.99%	https://archive.ics.uci.edu/ml/datasets/internet+advertisements
Apascal	12,695	64	1.38%	http://vision.cs.uiuc.edu/attributes/
Bank	41,188	62	11.26%	https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
Celeba	202,599	39	2.24%	http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
Census	299,285	500	6.20%	https://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29
Creditcard	284,807	29	0.17%	https://www.kaggle.com/mlg-ulb/creditcardfraud
Lung	145	3,312	4.13%	https://archive.ics.uci.edu/ml/datasets/Lung+Cancer
Probe	64,759	34	6.43%	http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
R8	3,974	9,467	1.28%	http://csmining.org/tl_files/Project_Datasets/r8_r52/r8-train-all-terms.txt
Secom	1,567	590	6.63%	https://archive.ics.uci.edu/ml/datasets/secom
U2R	60,821	34	0.37%	http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

R8, *20news*, *Sector* and *RCV1* are widely used text classification benchmark datasets. *Olivetti* is a widely-used face recognition dataset.

TABLE 5.7. Datasets used in the clustering task

Data	N	D	#Classes	Link
R8	7,674	17,387	8	http://csmining.org/tl_files/Project_Datasets/r8_r52/r8-train-all-terms.txt
20news	18,846	130,107	20	https://scikit-learn.org/0.19/datasets/twenty_newsgroups.html
Olivetti	400	4,096	40	https://scikit-learn.org/0.19/datasets/olivetti_faces.html
Sector	9,619	55,197	105	https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html#sector
RCV1	20,242	47,236	2	https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/binary.html#rcv1.binary

5.10 AUC-PR Performance of Ablation Study in Anomaly detection

The experimental results of AUC-PR performance of RDP and its variants in the anomaly detection task are shown in Table 5.8. Similar to the results shown in Table 5.3, using the L_{rdp} loss only, our proposed RDP model can achieve substantially better performance over its counterparts. By removing the L_{rdp} loss, the performance of RDP drops significantly in 11 out of 14 datasets. This demonstrates that the L_{rdp} loss is heavily harvested by our RDP model to learn high-quality representations from random distances. Removing L^{ad}_{aux} from RDP also results in substantial loss of AUC-PR in many datasets. This indicates both the random distance prediction loss L_{rdp} and the task-dependent loss L^{ad}_{aux} are critical to RDP. The boosting process is also important, but is not as critical as the two losses. Consistent with the observations derived from Table 5.3, distances calculated in non-linear and linear random mapping spaces are more effective supervisory sources than that in the original space.

TABLE 5.8. AUC-PR performance of RDP and its variants in the anomaly detection task.

	Decomposition				**Supervision Signal**	
Data	**RDP**	**RDP\L_{rdp}**	**RDP\L^{ad}_{aux}**	**RDP\Boosting**	**Org_SS**	**SRP_SS**
DDoS	0.301 ± 0.028	0.110 ± 0.015	0.364 ± 0.013	0.114 ± 0.001	0.363 ± 0.007	**0.380 ± 0.030**
Donors	**0.432 ± 0.061**	0.201 ± 0.033	0.104 ± 0.007	0.278 ± 0.040	0.099 ± 0.004	0.113 ± 0.010
Backdoor	0.305 ± 0.008	0.433 ± 0.015	0.142 ± 0.006	**0.537 ± 0.005**	0.143 ± 0.005	0.154 ± 0.028
Ad	**0.726 ± 0.007**	0.473 ± 0.009	0.491 ± 0.014	0.488 ± 0.008	0.419 ± 0.015	0.530 ± 0.007
Apascal	**0.042 ± 0.003**	0.021 ± 0.005	0.031 ± 0.002	0.028 ± 0.003	0.016 ± 0.003	0.035 ± 0.007
Bank	**0.364 ± 0.013**	0.258 ± 0.006	0.266 ± 0.018	0.278 ± 0.007	0.262 ± 0.016	0.265 ± 0.021
Celeba	**0.104 ± 0.006**	0.068 ± 0.010	0.060 ± 0.004	0.072 ± 0.008	0.050 ± 0.009	0.065 ± 0.010
Census	0.086 ± 0.001	0.081 ± 0.001	0.075 ± 0.001	**0.087 ± 0.001**	0.077 ± 0.002	0.064 ± 0.001
Creditcard	0.363 ± 0.011	0.290 ± 0.012	**0.414 ± 0.02**	0.329 ± 0.007	0.362 ± 0.016	0.372 ± 0.024
Lung	**0.705 ± 0.028**	0.381 ± 0.104	0.437 ± 0.083	0.542 ± 0.139	0.361 ± 0.054	0.464 ± 0.053
Probe	0.955 ± 0.002	0.609 ± 0.014	0.952 ± 0.007	0.628 ± 0.011	0.937 ± 0.005	**0.959 ± 0.011**
R8	0.146 ± 0.017	0.134 ± 0.031	0.109 ± 0.006	**0.173 ± 0.028**	0.067 ± 0.016	0.134 ± 0.019
Secom	**0.096 ± 0.001**	0.086 ± 0.002	0.096 ± 0.006	0.090 ± 0.001	0.088 ± 0.004	0.093 ± 0.004
U2R	0.261 ± 0.005	0.217 ± 0.011	**0.266 ± 0.007**	0.238 ± 0.009	0.187 ± 0.013	0.239 ± 0.023
#wins/draws/losses (RDP vs.)		13/0/1	11/0/3	11/0/3	12/0/2	5/0/9

5.11 NMI Performance of Ablation Study in Clustering

Table 5.9 shows the NMI performance of RDP and its variants in the clustering task. It is clear that our RDP model with the L_{rdp} loss is able to achieve NMI performance that is comparably well to the full RDP model, which is consistent to the observations in Table 5.5. Without using the L_{rdp} loss, the performance of the RDP model has

some large drops on nearly all the datasets. This reinforces the crucial importance of L_{rdp} to RDP, which also justifies that using L_{rdp} alone RDP can learn expressive representations. Similar to the results in Table 5.5, RDP is generally more reliable supervisory sources than Org_SS and SRP_SS in this set of results.

TABLE 5.9. NMI performance of RDP and its variants in the clustering task.

	Decomposition			Supervision Signal	
Data	**RDP**	**RDP**$\backslash L_{rdp}$	**RDP**$\backslash L_{aux}^{clu}$	**Org_SS**	**SRP_SS**
R8	0.539 ± 0.040	0.471 ± 0.043	0.505 ± 0.037	0.567 ± 0.021	**0.589 ± 0.039**
20news	**0.084 ± 0.005**	0.075 ± 0.006	0.081 ± 0.002	0.075 ± 0.002	0.074 ± 0.003
Olivetti	**0.805 ± 0.012**	0.782 ± 0.010	0.784 ± 0.010	0.795 ± 0.011	0.787 ± 0.011
Sector	0.305 ± 0.007	0.253 ± 0.010	**0.340 ± 0.007**	0.295 ± 0.009	0.298 ± 0.008
Rcv1	0.165 ± 0.000	0.146 ± 0.010	**0.168 ± 0.000**	0.154 ± 0.002	0.147 ± 0.000

5.12 Sensitivity w.r.t. the Dimensionality of Representation Space

This section presents the performance of RDP using different representation dimensions in its feature learning layer. The sensitivity test is performed for both anomaly detection and clustering tasks.

5.12.1 Sensitivity Test in Anomaly Detection

Figures 5.2 and 5.3 respectively show the AUC-ROC and AUC-PR performance of RDP using different representation dimensions on all the 14 anomaly detection datasets used in this work. It is clear from both performance measures that RDP generally performs stably w.r.t. the use of different representation dimensions on diverse datasets. This demonstrates the general stability of our RDP method on different application domains. On the other hand, the flat trends also indicate that, as an unsupervised learning source, the random distance cannot provide sufficient supervision information to learn richer and more complex representations in a higher-dimensional space. This also explains the performance on quite a few datasets where the performance of RDP decreases when increasing the representation dimension. In general, the representation dimension 50 is recommended for RDP to achieve effective anomaly detection on datasets from different domains.

5.12.2 Sensitivity Test in Clustering

Figure 5.4 presents the NMI and F-score performance of RDP-enabled K-means clustering using different representation dimensions on all the five datasets in the clustering task. Similar to the sensitivity test results in the anomaly detection task, on all the five datasets, K-means clustering performs stably in the representation space resulted by RDP with different representation dimensions. The clustering performance may drop a bit when the representation dimension is relatively low, e.g., 512. Increasing the representation to 1,280 may help RDP gain better representation power in some

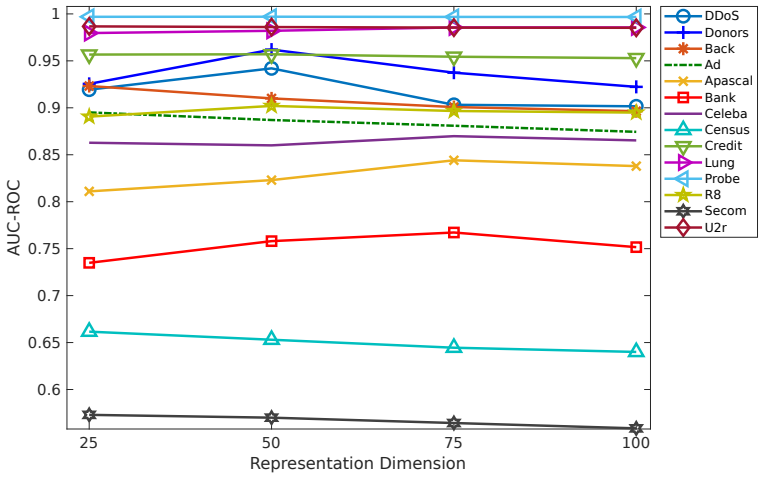

FIGURE 5.2. AUC-ROC results of RDP w.r.t. different representation dimensions on 14 datasets.

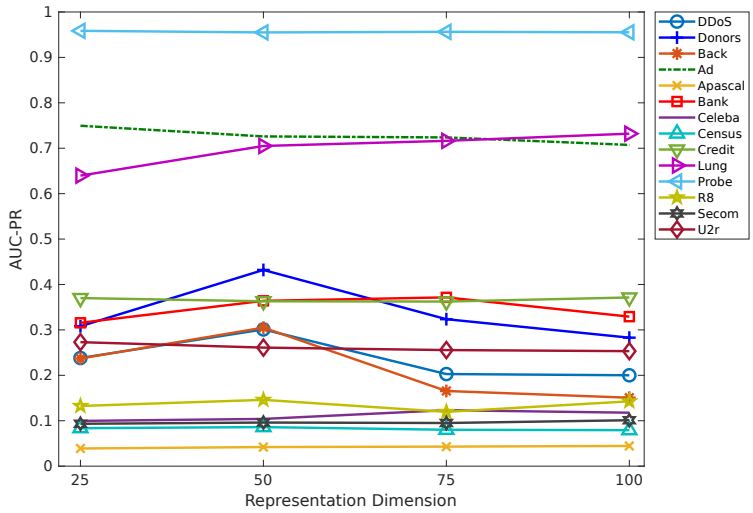

FIGURE 5.3. AUC-PR results of RDP w.r.t. different representation dimensions on 14 datasets.

datasets but is not a consistently better choice. Thus, the representation dimension 1,024 is generally recommended for clustering. Recall that the required representation dimension in clustering is normally significantly higher than that in anomaly detection, because clustering generally requires significantly more information to perform well than anomaly detection.

5.13 Computational Efficiency

The runtime of RDP is compared with its competing methods in both anomaly detection and clustering tasks. Since training time can vary significantly using different

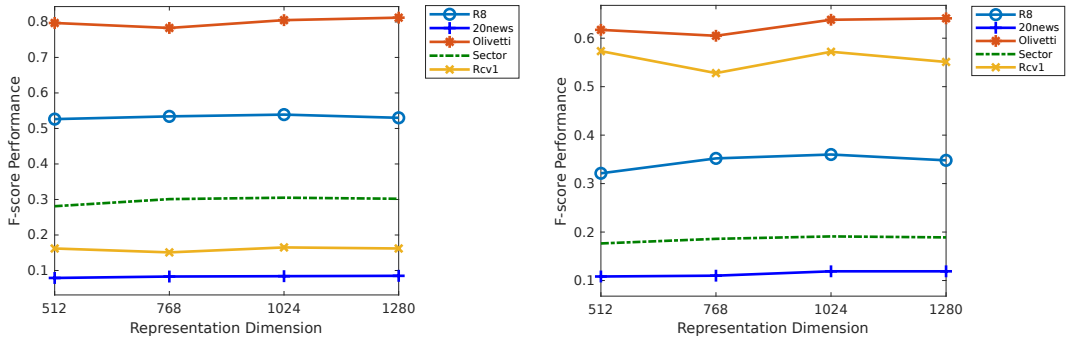

FIGURE 5.4. NMI and F-score performance of RDP-enabled K-means using different representation dimensions on all the five datasets used in clustering.

training strategies in deep learning-based methods, it is difficult to have a fair comparison of the training time. Moreover, the models can often be trained offline. Thus, we focus on comparing the runtime at the testing stage. All the runtime experiments below were done on a computing server node equipped with 32 Intel Xeon E5-2680 CPUs (2.70GHz) and 128GB Random Access Memory.

5.13.1 Testing Runtime in Anomaly Detection

The testing runtime in seconds of RDP and its five competing anomaly detection methods on 14 anomaly detection datasets are provided in Table 5.10. Since most of the methods integrate representation learning and anomaly detection into a single framework, the runtime includes the execution time of feature learning and anomaly detection for all six methods. In general, on most large datasets, RDP runs comparably fast to the most efficient methods iForest and RND, and is faster the two recently proposed deep methods REPEN and DAGMM. Particularly, RDP runs faster than REPEN and DAGMM by a factor of around five on high-dimensional and large-scale datasets like Donors and Census. RDP is slower than the competing methods in processing small datasets. This is mainly because RDP has a base runtime of its boosting process. Therefore, the runtime of RDP seems to be almost constant across the datasets. This is a very desired property for handling high-dimensional and large-scale datasets.

5.13.2 Testing Runtime in Clustering

Table 5.11 shows the testing runtime of RDP and its four competing methods in enabling clustering on five datasets. Since exactly the same K-means clustering is used on the features in all the five cases, we exclude the runtime of the K-means clustering for more straightforward comparison. The results show that RDP runs comparably fast to the very efficient methods SRP and AE since they do not involve complex computation at the testing stage; RDP runs about five orders of magnitude faster than HLLE since HLLE takes a huge amount of time in its nearest neighbours

TABLE 5.10. Testing runtime (in seconds) on 14 anomaly detection datasets.

Data Characteristics			RDP and Its Five Competing Methods					
Data	**N**	**D**	**iForest**	**AE**	**REPEN**	**DAGMM**	**RND**	**RDP**
DDoS	464,976	66	54.06	86.86	172.47	197.85	31.86	28.93
Donors	619,326	10	28.17	52.31	226.14	194.45	44.31	36.84
Backdoor	95,329	196	26.51	51.66	36.43	187.61	12.26	29.95
Ad	3,279	1,555	6.71	14.71	3.24	31.54	8.12	30.83
Apascal	12,695	64	6.53	4.27	6.30	69.35	3.62	22.88
Bank	41,188	62	9.72	6.87	17.25	170.56	9.31	28.47
Celeba	202,599	39	20.54	26.70	71.60	223.70	18.05	33.91
Census	299,285	500	155.77	225.29	121.08	236.21	42.83	57.74
Creditcard	284,807	29	22.45	29.38	103.18	235.93	20.97	30.84
Lung	145	3,312	6.20	13.11	2.16	39.75	1.44	24.29
Probe	64,759	34	9.55	10.06	28.14	131.40	9.90	29.61
R8	3,974	9,467	59.70	45.48	7.81	31.99	8.26	14.33
Secom	1,567	590	7.32	5.78	2.83	18.22	3.23	22.52
U2R	60,821	34	8.95	9.38	26.55	185.88	9.90	28.10

searching. Note that 'Org' indicates the clustering performed on the original space, so it involves no feature learning and does not take any time.

TABLE 5.11. Testing runtime (in seconds) on five clustering datasets.

Data Characteristics			RDP and Its Four Competing Methods				
Data	**N**	**D**	**Org**	**HLLE**	**SRP**	**AE**	**RDP**
R8	7,674	17,387	-	9,658.85	1.16	1.08	0.89
20news	18,846	130,107	-	94,349.20	2.26	11.49	6.85
Olivetti	400	4,096	-	166.02	0.73	0.03	0.03
Sector	9,619	55,197	-	24,477.80	1.40	4.28	2.87
RCV1	20,242	47,236	-	47,584.79	2.80	8.91	5.04

5.14 Comparison to State-of-the-art Representation Learning Methods for Raw Text and Image Data

Since RDP relies on distance information as its supervisory signal, one interesting question is that, can RDP still work when the presented data is raw data in a non-Euclidean space, such as raw text and image data? One simple and straightforward way to enable RDP to handle those raw data is, as what we did on the text and image data used in the evaluation of clustering, to first convert the raw texts/images into feature vectors using commonly-used methods, e.g., TF-IDF (Aizawa, 2003) for text data and treating each pixel as a feature unit for image data, and then perform RDP on these vector spaces. A further question is that, do we need RDP in handling those data since there are now a large number of advanced representation learning methods that are specifically designed for raw text/image datasets? Or, how is the performance of RDP compared to those advanced representation learning methods for raw text/image datasets? This section provides some preliminary results in the clustering task for answering these questions.

5.14.1 On Raw Text Data

On the raw text datasets R8 and 20news, we first compare RDP with the advanced document representation method Doc2Vec[3] as in (Le and Mikolov, 2014). Recall that, for RDP, we first use the bag-of-words model and document frequency information (e.g., TF-IDF) to simply convert documents into high-dimensional feature vectors and then perform RDP using the feature vectors. Doc2Vec leverages the idea of distributed representations to directly learn representations of documents. We further derive a variant of RDP, namely Doc2Vec+RDP, which performs RDP on the Doc2Vec projected representation space rather than the bag-of-words vector space. All RDP, Doc2Vec and Doc2Vec+RDP project data onto a 1,024-dimensional space for the subsequent learning tasks. Note that, for the method Doc2Vec+RDP, to better examine the capability of RDP in exploiting the Doc2Vec projected space, we first use Doc2Vec project raw text data onto a higher-dimensional space (5,120 dimensions for R8 and 10,240 dimensions for 20news), and RDP further learns a 1,024-dimensional space from this higher-dimensional space.

The comparison results are shown in Table 5.12. Two interesting observations can be seen. First, RDP can significantly outperform Doc2Vec on *R8* or performs comparably well on *20news*. This may be due to the fact that the local proximity information learned in RDP is critical to clustering; although the word prediction approach in Doc2Vec helps learn semantic-rich representations for words/sentences/-paragraphs, the pairwise document distances may be less effective than RDP since Doc2Vec is not like RDP that is designed to optimise this proximity information. Second, Doc2Vec+RDP can achieve substantially better performance than Doc2Vec, especially on the dataset *20news* where Doc2Vec+RDP achieves a NMI score of 0.198 while that of Doc2Vec is only 0.084. This may be because, as discussed in Section 5.5.3, RDP is equivalent to learn an optimised feature space out of its input space (Doc2Vec projected feature space in this case) using imperfect supervision information. When there is sufficient accurate supervision information, RDP can learn a substantially better feature space than its input space. This is also consistent with the results in Table 5.4, in which clustering based on the RDP projected space also performs substantially better than that working in the original space 'Org'.

5.14.2 On Raw Image Data

On the raw image dataset *Olivetti*, we compare RDP with the advanced representation learning method for raw images, RotNet (Gidaris, Singh, and Komodakis, 2018). RDP uses each image pixel as a feature unit and performs on a 64×64 vector space. RotNet directly learns representations of images by predicting whether a given image is rotated or not. Similar to the experiments on raw text data, we also evaluate the performance of RDP working on the RotNet projected space, i.e., RotNet+RDP.

[3] We use the implementation of Doc2Vec in a popular text mining python package `gensim` available at https://radimrehurek.com/gensim/index.html

TABLE 5.12. NMI and F-score performance of K-means clustering using RDP, Doc2Vec, and Doc2Vec+RDP based feature representations of the text datasets R8 and news20.

Data Characteristics			**NMI Performance**		
Data	**N**	**D**	**Doc2Vec**	**RDP**	**Doc2Vec+RDP**
R8	7,674	17,387	0.241 ± 0.022	**0.539 ± 0.040**	0.250 ± 0.003
20news	18,846	130,107	0.080 ± 0.003	0.084 ± 0.005	**0.198 ± 0.009**
Data Characteristics			**F-score Performance**		
Data	**N**	**D**	**Doc2Vec**	**RDP**	**Doc2Vec+RDP**
R8	7,674	17,387	0.317 ± 0.014	**0.360 ± 0.055**	0.316 ± 0.007
20news	18,846	130,107	0.115 ± 0.006	0.119 ± 0.006	**0.126 ± 0.009**

All RDP, RotNet and RotNet+RDP first learn a 1,024 representation space, and then K-means is applied to the learned space to perform clustering. In the case of RotNet+RDP, the raw image data is first projected onto a 2,048-dimensional space, and then RDP is applied to this higher-dimensional space to learn a 1,024-dimensional representation space.

We use the implementation of RotNet released by its authors[4]. Note that the original RotNet is applied to large image datasets and has a deep network architecture, involving four convolutional blocks with three convolutional layers for each block. We found directly using the original architecture is too deep for *Olivetti* and performs ineffectively as the data contains only 400 image samples. Therefore, we simplify the architecture of RotNet and derive four variants of RotNet, including $\text{RotNet}_{4\times2}$, $\text{RotNet}_{4\times1}$, $\text{RotNet}_{3\times1}$ and $\text{RotNet}_{2\times1}$. Here $\text{RotNet}_{a\times b}$ represents RotNet with a convolutional blocks and b convolutional layers for each block. Note that $\text{RotNet}_{2\times1}$ is the simplest variant we can derive that works effectively. We evaluate the original RotNet, its four variants and the combination of these five RotNets and RDP.

TABLE 5.13. NMI and F-score performance of K-means clustering using RDP, RotNet, and RotNet+RDP based feature representations of the image dataset Olivetti.

	NMI Performance	F-score Performance
Org	0.778 ± 0.014	0.590 ± 0.029
RDP	**0.805 ± 0.012**	**0.638 ± 0.026**
RotNet	0.467 ± 0.014	**0.243 ± 0.014**
RotNet+RDP	**0.472 ± 0.011**	0.242 ± 0.011
$\text{RotNet}_{4\times2}$	**0.518 ± 0.010**	0.281 ± 0.014
$\text{RotNet}_{4\times2}$+RDP	0.517 ± 0.010	**0.282 ± 0.014**
$\text{RotNet}_{4\times1}$	0.519 ± 0.010	0.283 ± 0.014
$\text{RotNet}_{4\times1}$+RDP	**0.536 ± 0.010**	**0.298 ± 0.011**
$\text{RotNet}_{3\times1}$	0.526 ± 0.014	0.303 ± 0.018
$\text{RotNet}_{3\times1}$+RDP	**0.567 ± 0.010**	**0.336 ± 0.015**
$\text{RotNet}_{2\times1}$	0.561 ± 0.010	0.339 ± 0.016
$\text{RotNet}_{2\times1}$+RDP	**0.587 ± 0.009**	**0.374 ± 0.015**

The evaluation results are presented in Table 5.13. Impressively, RDP can significantly

[4]The released code of RotNet is available at https://github.com/gidariss/FeatureLearningRotNet.

outperform RotNet and all its four variants on *Olivetti*. It is interesting that Org (i.e., performing K-means clustering on the original 64×64 vector space) also obtains a similar superiority over the RotNet family. This may be because *Olivetti* is too small to provide sufficient training samples for RotNet and its variants to learn its underlying semantic abstractions. This conjecture can also explain the increasing performance of RotNet variants with decreasing complexity of the RotNet architecture. Similar to the results on the raw text data, applying RDP on the RotNet projected spaces can also learn substantially more expressive representations than the representations yielded by RotNet and its variants, especially when the RotNet methods work well, such as the two cases: $\text{RotNet}_{3\times1}$ vs. $\text{RotNet}_{3\times1}$+RDP and $\text{RotNet}_{2\times1}$ vs. $\text{RotNet}_{2\times1}$+RDP.

5.15 Performance Evaluation in Classification

We also performed some preliminary evaluation of the learned representations in classification tasks using a feed-forward three-layer neural network model as the classifier. We used the same datasets as in the clustering task. Specifically, the representation learning model first outputs the new representations of the input data, and then the classifier performs classification on the learned representations. RDP is compared with the same competing methods HLLE, SRP, AE and COP as in clustering. F-score is used as the performance evaluation metric here.

The results are shown in Table 5.14. Similar to the performance in clustering and anomaly detection, our model using only the random distance prediction loss L_{rdp}, i.e., RDP$\backslash L_{aux}^{clu}$, performs very favourably and stably on all the five datasets. The incorporation of $\backslash L_{aux}^{clu}$ into the model, i.e., RDP, helps gain some extra performance improvement on datasets like *20news*, but it may also slightly downgrade the performance on other datasets. An extra hyperparameter may be added to control the importance of these two losses.

Table 5.14. F-score performance of classification on five real-world datasets.

Data	HLLE	SRP	AE	COP	RDP$\backslash L_{aux}^{clu}$	RDP
R8	0.246	0.895	0.874	0.860	0.900	**0.906**
20news	0.005	0.733	0.709	0.718	0.735	**0.753**
Olivetti	0.895	0.899	0.820	0.828	**0.900**	0.896
Sector	0.037	0.671	0.645	0.689	0.690	**0.696**
RCV1	0.766	0.919	0.918	N/A	**0.940**	0.926

Chapter 6

Conclusion

In this thesis, we have proposed a series of novel solutions for different tasks based on the idea of deep reinforcement learning. More specifically, it can be shown in the following three parts:

A novel traffic signal control dataset and a strong baseline. In order to alleviate the traffic congestion problem in urban areas, we propose a DRL model with an edge-weighted graph convolutional encoder and a unified structure decoder. In addition, we also propose a new dataset and settings along with the DRL model, including both synthetic and real traffic data in more complex scenarios.

Soft expert reward learning for vision-and-language navigation. Behaviour cloning based techniques directly copy the experts' behaviours and it will inevitably incur worse performance on the inference phase due to error accumulation. Reinforcement learning based methods have better generalisation ability. However, designing good reward functions is non-trivial. Therefore, we propose a generic soft expert reward learning strategy for the Vision-and-Language Navigation task. With the soft expert reward learning module, agents can distil the experts' policy directly without explicitly specifying the reward function by comparing the state-behaviour pair of agents and experts in a latent space. Along with the soft expert reward learning module, a self perceiving module targeting pushing the agent towards the final destination as fast as possible is further proposed. The experimental results demonstrate the effectiveness of the proposed method.

Unsupervised representation learning by predicting random distances. Finally, we apply the random network prediction to anomaly detection and clustering tasks in an unsupervised learning manner, inspired by the idea of random network prediction within deep reinforcement learning. We propose to learn important features without using any labelled data by training neural networks to predict data distances in a randomly projected space. In order to well predict the random distances, the representation learner is forced to learn the structures of data.

Future work. Despite remarkable achievements that have been gained by DRL models, nevertheless, there are many difficulties that are faced by Deep Reinforcement

Learning and its applications. Training unstable and insufficient sample efficiency are the two major issues that hamper DRL to be applied in many applications practically. Training unstable is caused by the random nature inside reinforcement learning, while comes with the sample efficiency issue. Targeting these two problems, how to seamlessly merge model-based RL models and model-free RL models is a potential direction for RL algorithm improvement. Moreover, depending on the natures of different applications, reward delaying and credit assignment are the other problems faced by DRL models, which could potentially be solved by attention models to perceive the "true" reward for each action. In the future, more efforts could be spent on designing a more generic and robust reward function framework to solve the aforementioned fundamental RL problems.

Bibliography

Agachi, Paul Serban, Mircea Vasile Cristea, Alexandra Ana Csavdari, and Botond Szilagyi (2016). *2. Model predictive control*. De Gruyter.

Agrawal, Pulkit, Joao Carreira, and Jitendra Malik (2015). "Learning to see by moving". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 37–45.

Agrawal, Shipra and Navin Goyal (2012). "Analysis of thompson sampling for the multi-armed bandit problem". In: *Conference on learning theory*. JMLR Workshop and Conference Proceedings, pp. 39–1.

Aizawa, Akiko (2003). "An information-theoretic perspective of tf–idf measures". In: *Information Processing & Management* 39.1, pp. 45–65.

Anderson, Peter, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. (2018a). "On evaluation of embodied navigation agents". In: *arXiv preprint arXiv:1807.06757*.

Anderson, Peter, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel (2018b). "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683.

Arel, Itamar, Cong Liu, T Urbanik, and AG Kohls (2010). "Reinforcement learning-based multi-agent system for network traffic signal control". In: *IET Intelligent Transport Systems* 4.2, pp. 128–135.

Arriaga, Rosa I and Santosh Vempala (1999). "An algorithmic theory of learning: Robust concepts and random projection". In: *40th Annual Symposium on Foundations of Computer Science*. IEEE, pp. 616–623.

Auer, Peter (2002). "Using confidence bounds for exploitation-exploration trade-offs". In: *Journal of Machine Learning Research* 3.Nov, pp. 397–422.

Baluja, Shumeet, Michele Covell, and Rahul Sukthankar (2017). "Traffic Lights with Auction-Based Controllers: Algorithms and Real-World Data". In: *CoRR* abs/1702.01205. arXiv: 1702.01205. URL: http://arxiv.org/abs/1702.01205.

Battaglia, Peter W, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. (2018). "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261*.

Beyer, Kevin, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft (1999). "When is "nearest neighbor" meaningful?" In: *International Conference on Database Theory*. Springer, pp. 217–235.

Bharucha-Reid, Albert T (1997). *Elements of the Theory of Markov Processes and their Applications*. Courier Corporation.

Billingsley, Patrick (1961). "Statistical methods in Markov chains". In: *The Annals of Mathematical Statistics*, pp. 12–40.

Bingham, Ella and Heikki Mannila (2001). "Random projection in dimensionality reduction: applications to image and text data". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 245–250.

Britannica (2006). "Markov process. https://www.britannica.com/science/Markov-process". In: *Encyclopedia Britannica*.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). "Openai gym". In: *arXiv preprint arXiv:1606.01540*.

Bronstein, Michael M, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst (2017). "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4, pp. 18–42.

Burda, Yuri, Harrison Edwards, Amos Storkey, and Oleg Klimov (2019). "Exploration by random network distillation". In: *ICLR*.

Chen, Xi, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel (2016). "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2172–2180.

Chu, Tianshu, Jie Wang, Lara Codecà, and Zhaojian Li (2019). "Multi-agent deep reinforcement learning for large-scale traffic signal control". In: *IEEE Transactions on Intelligent Transportation Systems* 21.3, pp. 1086–1095.

Covell, Michele, Shumeet Baluja, and Rahul Sukthankar (2015). "Micro-auction-based traffic-light control: Responsive, local decision making". In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, pp. 558–565.

Demšar, Janez (2006). "Statistical comparisons of classifiers over multiple data sets". In: *Journal of Machine learning research* 7.Jan, pp. 1–30.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee, pp. 248–255.

Denker, John S, WR Gardner, Hans Peter Graf, Donnie Henderson, Richard E Howard, W Hubbard, Lawrence D Jackel, Henry S Baird, and Isabelle Guyon (1989). "Neural network recognizer for hand-written zip code digits". In: *Advances in neural information processing systems*. Citeseer, pp. 323–331.

Doersch, Carl, Abhinav Gupta, and Alexei A Efros (2015). "Unsupervised visual representation learning by context prediction". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430.

Donoho, David L and Carrie Grimes (2003). "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data". In: *Proceedings of the National Academy of Sciences* 100.10, pp. 5591–5596.

Fried, Daniel, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell (2018). "Speaker-follower models for vision-and-language navigation". In: *Advances in Neural Information Processing Systems*, pp. 3314–3325.

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, pp. 267–285.

Genders, Wade and Saiedeh Razavi (2016). "Using a deep reinforcement learning agent for traffic signal control". In: *arXiv preprint arXiv:1611.01142*.

Gidaris, Spyros, Praveer Singh, and Nikos Komodakis (2018). "Unsupervised representation learning by predicting image rotations". In: *ICLR*.

Goodfellow, Ian J, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial networks". In: *arXiv preprint arXiv:1406.2661*.

Goroshin, Ross, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun (2015). "Unsupervised learning of spatiotemporally coherent metrics". In: *Proceedings of the IEEE international conference on computer vision*, pp. 4086–4093.

Guo, Mengyu, Pin Wang, Ching-Yao Chan, and Sid Askary (2019). "A Reinforcement Learning Approach for Intelligent Traffic Signal Control at Urban Intersections". In: *arXiv preprint arXiv:1905.07698*.

Ha, David and Jürgen Schmidhuber (2018). "World models". In: *arXiv preprint arXiv:1803.10122*.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning*. PMLR, pp. 1861–1870.

Hartigan, John A and Manchek A Wong (1979). "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1, pp. 100–108.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Heess, Nicolas, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. (2017). "Emergence of locomotion behaviours in rich environments". In: *arXiv preprint arXiv:1707.02286*.

Hill, Felix, Kyunghyun Cho, and Anna Korhonen (2016). "Learning distributed representations of sentences from unlabelled data". In: *15th Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1367–1377.

Hinton, Geoffrey E and Sam T Roweis (2003). "Stochastic neighbor embedding". In: *Advances in neural information processing systems*, pp. 857–864.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786, pp. 504–507.

Ho, Jonathan and Stefano Ermon (2016). "Generative adversarial imitation learning". In: *Advances in neural information processing systems*, pp. 4565–4573.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Johnson, William B and Joram Lindenstrauss (1984). "Extensions of Lipschitz mappings into a Hilbert space". In: *Contemporary mathematics* 26.189-206, p. 1.

Ke, Liyiming, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa (2019). "Tactical rewind: Self-correction via backtracking in vision-and-language navigation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6741–6749.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907*.

Konda, Vijay R and John N Tsitsiklis (2000). "Actor-critic algorithms". In: *Advances in neural information processing systems*. Citeseer, pp. 1008–1014.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25, pp. 1097–1105.

Le, Quoc and Tomas Mikolov (2014). "Distributed representations of sentences and documents". In: *International conference on machine learning*, pp. 1188–1196.

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4, pp. 541–551.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lee, Hsin-Ying, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang (2017). "Unsupervised representation learning by sorting sequences". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 667–676.

Levine, Sergey and Vladlen Koltun (2013). "Guided policy search". In: *International conference on machine learning*. PMLR, pp. 1–9.

Li, Li, Ding Wen, and Danya Yao (2014). "A survey of traffic control with vehicular communications". In: *IEEE Transactions on Intelligent Transportation Systems* 15.1, pp. 425–432.

Li, Ping, Trevor J Hastie, and Kenneth W Church (2006). "Very sparse random projections". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 287–296.

Liang, Xiaoyuan, Xunsheng Du, Guiling Wang, and Zhu Han (2018). "Deep reinforcement learning for traffic light control in vehicular networks". In: *arXiv preprint arXiv:1803.11115*.

Lillicrap, Timothy P, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2015). "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971*.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008). "Isolation forest". In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE, pp. 413–422.

Liu, Xiao-Yang, Zihan Ding, Sem Borst, and Anwar Walid (2018). "Deep Reinforcement Learning for Intelligent Transportation Systems". In: *arXiv preprint arXiv:1812.00979*.

Lopez, Pablo Alvarez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner (2018). "Microscopic Traffic Simulation using SUMO". In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE. URL: https://elib.dlr.de/124092/.

Ma, Chih-Yao, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong (2019a). "Self-monitoring navigation agent via auxiliary progress estimation". In: *arXiv preprint arXiv:1901.03035*.

Ma, Chih-Yao, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira (2019b). "The regretful agent: Heuristic-aided navigation through progress estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6732–6740.

Mannion, Patrick, Jim Duggan, and Enda Howley (2016). "An experimental review of reinforcement learning algorithms for adaptive traffic signal control". In: *Autonomic Road Transport Support Systems*. Springer, pp. 47–66.

McInnes, Leland, John Healy, and James Melville (2018). "UMAP: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426*.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013a). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013b). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Misra, Ishan, C Lawrence Zitnick, and Martial Hebert (2016). "Shuffle and learn: unsupervised learning using temporal order verification". In: *European Conference on Computer Vision*. Springer, pp. 527–544.

Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous

methods for deep reinforcement learning". In: *International conference on machine learning*, pp. 1928–1937.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, p. 529.

Mousavi, Seyed Sajad, Michael Schukat, and Enda Howley (2017). "Traffic light control using deep policy-gradient and value-function-based reinforcement learning". In: *IET Intelligent Transport Systems* 11.7, pp. 417–423.

Ng, Andrew Y, Stuart J Russell, et al. (2000). "Algorithms for inverse reinforcement learning." In: *Icml*. Vol. 1, pp. 663–670.

Nishi, Tomoki, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura (2018). "Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 877–883.

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748*.

Pang, Guansong, Longbing Cao, Ling Chen, and Huan Liu (2018). "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, pp. 2041–2050.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic differentiation in pytorch". In.

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). "Curiosity-driven exploration by self-supervised prediction". In: *International Conference on Machine Learning*. PMLR, pp. 2778–2787.

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros (2016). "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544.

Pham, Tong Thanh, Tim Brys, Matthew E Taylor, Tim Brys, Madalina M Drugan, PA Bosman, Martine-De Cock, Cosmin Lazar, L Demarchi, David Steenhoff, et al. (2013). "Learning coordinated traffic light control". In: *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*. Vol. 10. IEEE, pp. 1196–1201.

Racanière, Sébastien, Théophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. (2017). "Imagination-augmented agents for deep reinforcement learning". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5694–5705.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *ICLR*.

Rahimi, Ali and Benjamin Recht (2008). "Random features for large-scale kernel machines". In: *Advances in neural information processing systems*, pp. 1177–1184.

Rahmani, Mostafa and George Atia (2017). "Coherence pursuit: Fast, simple, and robust subspace recovery". In: *Proceedings of the 34th International Conference on Machine Learning*. JMLR. org, pp. 2864–2873.

Reddy, Siddharth, Anca D Dragan, and Sergey Levine (2019). "SQIL: imitation learning via regularized behavioral cloning". In: *arXiv preprint arXiv:1905.11108*.

Richter, S (2006). "Learning road traffic control: towards practical traffic control using policy gradients". In: *Albert-Ludwigs-Universitat Freiburg, Fakultat fur Angewandte Wissenschaften, Institut fur Informatik, Germany*.

Richter, Silvia, Douglas Aberdeen, and Jin Yu (2007). "Natural actor-critic for road traffic optimisation". In: *Advances in neural information processing systems*, pp. 1169–1176.

Ritcher, S (2007). "Traffic light scheduling using policy-gradient reinforcement learning". In: *The International Conference on Automated Planning and Scheduling., ICAPS*.

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell (2011). "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 627–635.

Roweis, Sam T and Lawrence K Saul (2000). "Nonlinear dimensionality reduction by locally linear embedding". In: *Science* 290.5500, pp. 2323–2326.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Rummery, Gavin A and Mahesan Niranjan (1994). *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK.

Sallab, Ahmad EL, Mohammed Abdou, Etienne Perot, and Senthil Yogamani (2017). "Deep reinforcement learning framework for autonomous driving". In: *Electronic Imaging* 2017.19, pp. 70–76.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2009). "The graph neural network model". In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.

Schaul, Tom, John Quan, Ioannis Antonoglou, and David Silver (2015). "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952*.

Schepperle, Heiko and Klemens Böhm (2008). "Auction-based traffic management: Towards effective concurrent utilization of road intersections". In: *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*. IEEE, pp. 105–112.

Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller (1997). "Kernel principal component analysis". In: *International conference on artificial neural networks*. Springer, pp. 583–588.

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (2015). "Trust region policy optimization". In: *International conference on machine learning*. PMLR, pp. 1889–1897.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347*.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *The Bell system technical journal* 27.3, pp. 379–423.

Silver, David (2015). "Introduction to Reinforcement Learning. https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver". In: *DeepMind*.

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. (2017). "Mastering the game of go without human knowledge". In: *nature* 550.7676, pp. 354–359.

Sun, Wen, Arun Venkatraman, Geoffrey J Gordon, Byron Boots, and J Andrew Bagnell (2017). "Deeply aggrevated: Differentiable imitation learning for sequential prediction". In: *International Conference on Machine Learning*. PMLR, pp. 3309–3318.

Sutton, Richard S (1991). "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM Sigart Bulletin* 2.4, pp. 160–163.

Sutton, Richard S, David A McAllester, Satinder P Singh, Yishay Mansour, et al. (1999). "Policy gradient methods for reinforcement learning with function approximation." In: *NIPs*. Vol. 99. Citeseer, pp. 1057–1063.

Tan, Hao, Licheng Yu, and Mohit Bansal (2019). "Learning to navigate unseen environments: Back translation with environmental dropout". In: *arXiv preprint arXiv:1904.04195*.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5026–5033.

Van Hasselt, Hado, Arthur Guez, and David Silver (2016). "Deep Reinforcement Learning with Double Q-Learning." In: *AAAI*. Vol. 2. Phoenix, AZ, p. 5.

Vempala, Santosh (1998). "Random projection: A new approach to VLSI layout". In: *Proceedings 39th Annual Symposium on Foundations of Computer Science*. IEEE, pp. 389–395.

Vermorel, Joannes and Mehryar Mohri (2005). "Multi-armed bandit algorithms and empirical evaluation". In: *European conference on machine learning*. Springer, pp. 437–448.

Vincent, Pascal, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11, pp. 3371–3408.

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782, pp. 350–354.

Wang, Ruohan, Carlo Ciliberto, Pierluigi Amadori, and Yiannis Demiris (2019a). "Random expert distillation: Imitation learning via expert policy support estimation". In: *arXiv preprint arXiv:1905.06750*.

Wang, Xiaoqiang, Liangjun Ke, Zhimin Qiao, and Xinghua Chai (2020). "Large-scale traffic signal control using a novel multiagent reinforcement learning". In: *IEEE transactions on cybernetics* 51.1, pp. 174–187.

Wang, Xin, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang (2019b). "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6629–6638.

Wang, Ziyu, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas (2016). "Dueling network architectures for deep reinforcement learning". In: *International conference on machine learning*. PMLR, pp. 1995–2003.

Wei, Hua, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li (2019). "Colight: Learning network-level cooperation for traffic signal control". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1913–1922.

Wold, Svante, Kim Esbensen, and Paul Geladi (1987). "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3, pp. 37–52.

Wu, Tong, Pan Zhou, Kai Liu, Yali Yuan, Xiumin Wang, Huawei Huang, and Dapeng Oliver Wu (2020). "Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks". In: *IEEE Transactions on Vehicular Technology* 69.8, pp. 8243–8256.

Yadav, Deshraj, Rishabh Jain, Harsh Agrawal, Prithvijit Chattopadhyay, Taranjeet Singh, Akash Jain, Shiv Baran Singh, Stefan Lee, and Dhruv Batra (2019). "EvalAI: Towards Better Evaluation Systems for AI Agents". In.

Zhang, Richard, Phillip Isola, and Alexei A Efros (2017). "Split-brain autoencoders: Unsupervised learning by cross-channel prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1058–1067.

Zhang, Ye and Byron Wallace (2015). "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1510.03820*.

Zhou, Tinghui, Matthew Brown, Noah Snavely, and David G Lowe (2017). "Unsupervised learning of depth and ego-motion from video". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1851–1858.

Zong, Bo, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen (2018). "Deep autoencoding gaussian mixture model for unsupervised anomaly detection". In: *ICLR*.