Dynamic Pricing for Predictive Analytics in Parking

by

Deyu Deng

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada

September 2021

Thesis advisor

Dr. Carson K. Leung

Author

Deyu Deng

Dynamic Pricing for
Predictive Analytics in Parking

Abstract

Despite urbanization benefiting modern society and the people living in the urban city, the limited public resources, especially parking resources, remain a problem. Parking pricing acts as a tool to adjust the available resources. A logical question is: How to use parking pricing to maximize parking resource utilization while optimizing the parking revenue for parking management? In this MSc thesis, I propose an architecture that utilizes available public resources while optimizing revenue with predefined restrictions, especially in the parking management field. More specifically, I first (a) design a data-driven time series based prediction model, and then (b) design a reinforcement learning based dynamic pricing model to incorporate price restrictions. Moreover, I also (c) come up with metrics to evaluate the dynamic pricing model, as well as (d) implement and evaluate the proposed models with real parking data. Evaluation results show the effectiveness and practicality of my predictive analytics architecture for dynamic pricing for parking applications.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Special thanks to my wife, my parents, my academic advisor (Dr. Carson K. Leung), my advisory/examining committee members (Dr. Mike Domaratzki from Computer Science at University of Manitoba/Western University, and Dr. Xikui Wang from Warren Centre for Actuarial Studies and Research in Asper School of Business), MSc defence chair (Dr. Parimala Thulasiraman), and also the people who have supported me along the way.

I also thank Mitacs and Winnipeg Airports Authority (WAA) for their financial support. Thanks also to Scott Marohn, Colin McFadyen, Ronalyn Olaes-Zimolag, Trevor Strome, Ran Wei, and Balder Zamorano at WAA, as well as Blake Podaima at Virtuistix Inc./Manitoba Training & Research Consortia (MTRC), for their domain expertise.

Besides, I also appreciate myself for holding up over days and nights, across fears and tears.

Deyu Deng
B.Sc. (Maj.), The University of Manitoba, 2017

The University of Manitoba

September 02, 2021

This thesis is dedicated to someone special.

I miss ya.

Chapter 1

Introduction

This is the era of artificial intelligence (AI) and data science. High volumes of data are being collected from numerous sources—sensors, payment transactions, traffic conditions. Afterwards, people try to understand the data they collected and make use of them. Data science has been purposed to deal with the challenge from the data. Different skills and areas under data science have been developed:

- Data mining aims to extract useful information and valuable insight such as patterns from large batches of data.
- AI, on the other hand, provides the ability to make decisions on machines by utilizing the "smart" algorithms along with the voluminous data.

In daily life, there are problems that can be enhanced with modern technologies, including AI. For example, the situation exists that the available resources may not be fully utilized, especially in the parking management field. On the one hand, it is a typical sensorial that in the urban city people can not find a parking lot easily

during the peak/rush hours. Drivers are circling around to find the possible parking lot while wasting time and fuel (or resources). On the other hand, from the parking management perspective, the parking space will be fully utilized on working days but most likely will in less use during the night due to the nature of the human activity.

Researchers have come up different approaches to utilize the parking resources with modern technologies. For example, Zou et al. [ZKWL15] proposed a pricing mechanism design for parking lot assignment in the information era.

In addition, the smartphone reservation system also has been being widely used in modern cities. For example, Sheelarani et al. [SASS16] discussed the parking reservation system with an Internet of Things (IoT) based Intelligent Parking System (IPS). It provides the ability of a user to use an Android application to book a reservation and make a payment. The researcher also noticed that drivers had different preferences on the types of parking, such as garage parking vs. curbside parking [IL15; ZZ16; DYMJ19].

AI has also been applied in this field. For example, Jiang et al. [JWW+18] applied big data analysis and neural networks to recognize the vehicle plate number, which is broadly used in the ticketless parking system.

There are only a limited number of researchers who considered the parking challenge from a parking revenue management perspective. Parking price as a tool, which can affect the modern traffic system [Sho17], could be used to adjust the demand for parking resources. Dynamic pricing, on the other hand, is a study that aims to optimize the selling price, has been involved as a method in this situation to balance the limited resource under dynamic demand. According to Tian et al.'s work [TYWH18],

the dynamic price could benefit revenue as well as the utilization of parking spaces. Lei et al. [LO17] developed approximate dynamic programming (ADP) based dynamic pricing policy with mathematical programming with equilibrium constraints (MPEC) model for IPS. The numerical outcome indicates it achieves better system performance.

However, there is a common condition that exists in real life. Some parking lot has a policy that specifies the boundary of the parking price. Also, it may restrict the amount of rate change for each time. For example, the policy may indicate the maximum allowed parking price; also, each price adjustment should be less than a certain amount. Since parking is a way of public service, it has to preset the maximum pricing, which can be treated as the 'ceiling' of the price. The 'ceiling' of the price is to guarantee that such a resource is affordable to the public. As a result, how to utilize the available parking resource so public resources will not be wasted while optimizing the parking revenue for parking management agencies with predefined 'ceiling' has become a challenge.

1.1 Thesis statement and contributions

In my thesis, I propose an architecture that utilizes the available public resource while optimizing revenue with predefined restrictions, especially on the parking management field. In particular, I investigate on the following questions:

Q1. As the available resource fluctuates year over year, can the system adapt itself to the resource fluctuations?

Q2. How can we distribute the limited resource to the people that urgently need the resource?

Q3. How to integrate different policy restrictions into the model (e.g., what if the policy regulates the final price should be less than a certain amount)?

Q4. After concatenating different components, what would be the best metrics to evaluate the overall consequence?

To answer these questions, I design a data-driven time series based prediction model and a dynamic pricing model based on reinforcement learning. I also adapt and optimize these two models with price restrictions. To measure the performance, I evaluate these models with real-life data. Hence, my *key contributions* include:

1. my design of a data-driven time series based prediction model,
2. my design of another dynamic pricing model based on reinforcement learning,
3. my adaptation and optimization of the aforementioned prediction and dynamic pricing models with price restrictions,
4. my design of evaluation metrics for the dynamic pricing model, and
5. my implementation and evaluation of these proposed models with real data.

This thesis is organized as follows. The next chapter introduces the related background techniques—especially the time series prediction and reinforcement learning. Chapter 3 provides the literature review for the dynamic pricing field. Chapter 4 introduces the details of this dynamic pricing architecture for predictive analytic in

parking. Chapter 5 discuss the experiment result for this thesis. Finally, Chapter 6 draws the conclusion and discusses future work.

Chapter 2

Background

As this thesis covers two sub-problems (time series prediction and dynamic pricing), I provide some background information on these two sub-problems in this chapter. In particular, for the time series prediction problem, I first describe different types of time series prediction and then mainly focus on two types—namely, auto-regressive integrated moving average (ARIMA) and neural network-based time series prediction. As for dynamic pricing, I describe dynamic pricing with reinforcement learning.

2.1 Time series prediction

Time series prediction (TSP) approximates a particular duration based on given sequential observations.

Definition 2.1.1 (Time Series Prediction [HZL$^+$21]) *By given a series of temporal related observations x_t, we expect to get the predicted result x_{t+n}. It can be*

expressed as:

$$x_{t+n} = f(x_t, \cdots, x_{t-h}) \tag{2.1}$$

where $n, h \geq 1$. *The* x_{t+n} *represents the predictions that can be one or multiple steps.*

2.1.1 Auto-regressive integrated moving average (ARIMA) model

The auto-regressive integrated moving average (ARIMA) model [Ham95] is the statistical approach that describes the behaviour of a noisy linear dynamical system. It can represent the time series due to its flexible modelling capability [LHZS16]. It was initially performed in econometrics by Box and Jenkins [BJ90].

In detail, ARIMA extends the existing auto-regressive moving average (ARMA) model by the integrated auto-regressive (AR) model with the moving average (MA) model. The general form of ARIMA model can be expressed as $ARIMA(p, q)$ with the following equation [STS18]:

$$x_t = c + \sum_{i=1}^{p}(\phi_i x_{t-i}) + \epsilon_t + \sum_{i=0}^{q}(\theta_i \epsilon_{t-i}) \tag{2.2}$$

where

- c is the constant.
- p represents that the AR model uses the dependencies between the observation and the number of lagged observations.
- ϕ_i indicates the auto-correlation coefficients, and $\phi_i \neq 0$.

- ϵ_t indicates the Gaussian white noise series.
- q represents the lagged observations of the forecast errors from the MA model.
- θ_i is a weight applied to the stochastic term in the time series where $\theta_i = 1$, and $\theta_i \neq 0$.

After differentiating the time series from a non-stationary time series into a stationary time series by the integrated step, the general form of an ARIMA model can be expressed as $ARIMA(p, d, q)$ [STS18].

ARIMA has been shown its success in multiple fields. For example, Benvenuto et al. [BGV+20] used the ARIMA model on Johns Hopkins epidemiological data to predict the coronavirus disease 2019 (COVID-19) trends. Arumugam and Saranya [AS18] used ARIMA as a statistical approach to making the rainfall prediction. Xu et al. [XQH10] utilized the ARIMA model to forecast the demand of commodities after natural disasters.

Despite its success, ARIMA also suffers from few limitations. For example, it is challenging for a simple ARIMA model to handle the nonlinear relationships between input variables [STS18]. Note that, in a real-world situation, the relationship between observations may not be linear. Moreover, existing algorithms for estimating parameters of ARIMA have to access the entire data set in advance, which may not adapt to the steaming characteristics of time series data [LHZS16].

2.1.2 Neural network-based time series prediction

Things have improved with new techniques, especially with the deep learning models. A simple neural network would enhance things differently.

2.1.3 Recurrent neural network (RNN)

The recurrent neural networks (RNN) are a kind of neural network that handles sequence data. Unlike the artificial neural network (ANN) (in which the neurons are all independent of each other), the RNN uses the output from the previous step as the input for the next step. The main idea of RNN is to make use of the sequential information from earlier stages. The reason it is called "recurrent" is that it follows the same logic to handle every element from a sequence. The internal state from RNN records the information from past input. However, in practice, the limitation of RNN is that it only remembers the information from a few steps earlier. RNN suffers from the vanishing gradient problem. As a result, RNN is not the best option for handling the long sequence data, especially the time series data.

2.1.4 Long short-term memory (LSTM)

The long short-term memory (LSTM), on the other hand, solves the drawback of the RNN. It is a particular type of RNN with extra components that can learn long-term sequences. The LSTM unit contains a cell, an input gate, an output gate and a forget gate. The cell is used to remember the values and connects the modules from one to another. The gates are the layers mainly used to control and adjust the content in the LSTM cell. In each gate layer, it produces numbers from the range of

0 to 1. Note that 0 indicates nothing will be passed while 1 means everything will be passed without modification.

The forget gate generates numbers between 0 and 1 that are mainly used to control what information will be ignored. The input gate uses the numbers to decide what values will be updated. The output gate follows similar patterns as other gates but in different behaviour. It uses the numbers to decide what values will be filtered.

The LSTM is capable of handling the long sequence data in practice. As a result, I choose LSTM on my thesis to make predictions.

2.2 Reinforcement learning

Reinforcement learning is the technique that acts as an agent to provide optimal actions based on a reward function under an environment. It differs from supervised learning in which the model is trained with the correct result. The reinforcement learning model has to learn the optimal action that it can take from the experience by utilizing the rewards. There are several terminologies in reinforcement learning:

- *Rewards* define how good or bad an action is taken. It is the core of reinforcement learning application. To better define a reinforcement learning problem, the reward has to be described by the maximization of expected cumulative reward. Once we have the reward, the overall goal aims to maximize the reward.

- An *agent* can be represented by the reinforcement learning models. It takes observations from the environment as input and provides the optimal actions defined by the policy.

- An *action* indicates all the possible behaviour that an agent can take under an environment. The action can interact with the environment and produce the state.

- *Environment* indicates the things that agents can interact with. It receives the action taken by the agent and provides the reward based on the reward function.

- *Q function*—aka action-value function—calculates the Q value, which measures the overall expected reward in a state with a specific action.

- A *state* indicates the relationship between action and the environment. It is used to decide what happens next.

- A *policy* maps state with action. It can be used to pick the optimal action for a state.

- The *Bellman equation* [SW10] is a method that is used to find the optimal solution for dynamic programming. It can transform the dynamic optimization problem into a sequence problem.

2.2.1 Markov decision process (MDP)

By combine all the aforementioned terms together, we can make a simple Markov Decision Process (MDP) denoted as $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where

- $\mathcal{S}$ is a set of states, which contains all the information to decide what will happen.

- $\mathcal{A}$ contains a set of actions agent can take.

- $\mathcal{P}$ is the transition probability matrix which defines the transition probabilities from all states s to their next state s'.
- $\mathcal{R}$ is the reward function
- γ where $\gamma \in [0, 1]$ is a factor to adjust focus in MDP.

Once we have the MDP, we can calculate the reward based on a reward function with state-value function along with Bellman equation as follows:

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi \left(S_{t+1} \right) \mid S_t = s \right] \tag{2.3}$$

where

- s indicates for the state
- $\mathbb{E}$ gives the expected value for policy π
- R_{t+1} represents to the immediate reward
- $\gamma v_\pi \left(S_{t+1} \right)$ represents the discounted value of other state.
  - γ indicates the discount factor
  - S_{t+1} means the state for time $t + 1$.
  - $v_\pi \left(S_{t+1} \right)$ represents the state value for state S_{t+1} at time $t + 1$.

To find the optimal value for maximizing the reward, the Bellman optimality equation is used:

$$V^*(s) = \max_\pi v_\pi(s) \tag{2.4}$$

where

- s indicates for the state
- $V^*(s)$ indicates the optimal value for state s
- π indicates the policy
- v indicates the value for the state s that calculated based on the Equation (2.3)

The idea of Equation (2.4) is to find the policy π that brings the maximum value v for the state s. The optimal value could used to solve the MDP. However, MDP might be high dimensional in some complex cases, and it is difficult to be resolved.

2.2.2 Q learning

In addition, Q-learning is another approach in reinforcement learning. It does not need a model. Instead, it utilizes the Q table, which contains the expected future rewards for actions that can be taken at each stage to find the optimal action that can be taken for a state.

In more details, the Q value (or the action-value) can be expressed by the equation:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s, a_t = a, \pi\right] \tag{2.5}$$

where

- $\mathbb{E}$ gives the expected value.
- r_t refers to the reward at time t.

- s indicates the state.
- a indicates the action.
- π indicates the policy.

Based on Equation (2.5), the Q value for time t can depend on the rewards from other times.

As a result, by combining Equation (2.5) with Bellman equation (i.e., Equation (2.4)), the optimal Q function can be calculated as follows:

$$Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*\left(s', a'\right)\right] \tag{2.6}$$

where

- $\mathbb{E}$ gives the expected maximum value
- r indicates the reward
- γ indicates to the discount factor
- $Q^*\left(s', a'\right)$ represents the maximum Q value for the state s' of the action a'

2.2.3 Deep reinforcement learning

Deep reinforcement learning, on the other hand, adapts deep learning with the traditional reinforcement learning concept. Deep Q learning(DQN) [MKS+15] is one of the most popular deep reinforcement learning approaches. In more detail, instead of finding the optimal Q value, deep Q learning uses neural networks as the function

approximator to estimate Q values with the Bellman optimally Equation (2.6). In more details, it uses three layers of convolutional neural network followed by a fully connected network to estimate the Q value. It also uses a technique called "Experience replay" to stabilize the DQN network.

2.3 Summary

In summary, the time series prediction has shown its power in forecasting related temporal observations. For instance, the traditional *ARIMA* is a time series prediction technique that handles time-series observations from a statistical approach but suffers from the nonlinear observations. On the other hand, *LSTM* has shown not only its ability to handle nonlinear observations but also its power in multivariate predictions. Moreover, as an important topic in machine learning, *reinforcement learning* mainly focuses on the interaction between the modelled agent and the preset environment. *Deep reinforcement learning*, which combines reinforcement learning with deep learning techniques, has become a widespread technique in the field. As a type of deep reinforcement learning algorithm, *deep Q learning* utilizes deep learning techniques as a function approximator with Q-learning to estimate the Q values to find the optimal actions for the state.

Chapter 3

Related works

Here, I discuss related works on two techniques (dynamic pricing and reinforcement learning), which are relevant to this thesis.

3.1 Dynamic pricing

The reach for dynamic pricing has been ongoing. For instance, Netessine and Shumsky [NS02] indicated the dynamic pricing strategy has been applied in different industries, such as airlines and hotels.

The purpose of dynamic pricing is to adjust the product price dynamically based on different factors, so that the product is sold to a specific customer on a particular time and cost, to earn the maximum revenue. Dynamic pricing is highly correlated to demand, inventory, and price history [Den15]. Among all factors, customer demand is the most crucial part. Demand could be uncertain or fixed. The inventory, acting as an available resource that sellers can provide to buyers, could also be limited

or unlimited. Price is also an essential part. The seller uses the price as a tool to manage revenue, while the buyers use price to evaluate the actual needs. In recent years, the dynamic pricing strategy has been widely applied in different fields with more enhancement on newer technologies. For example, Airbnb uses a customized regression model to take advantage of a binary classification model with the regression model, to predict user's booking action to optimize the price to maximize the revenue [YQC+18].

In recent years, with the new technologies such as the internet of things (IoT) used in the parking management field, the concept of 'smart parking' has been applied and continuously changed the parking management methods. Parking authority is able to know the real-time parking availability by using the internet of things. Online booking and availability checking were also developed in the past couple years.

The usage of newer technologies provides a dynamic management of parking resources, which then leads to dynamic pricing. For example, Zheng and Geroliminis [ZG16] developed an Macroscopic Fundamental Diagram (MFD) multi-modal traffic modeling approach to capture congestion at network level for car and bus and integrated with a parking model with pricing strategies.

Mackowski et al. [MBO15] tried to bring the concept from the Stackelberg leader-follower game model theories and converted it into a dynamic Mathematical Program Equilibrium Constraints (MPECs). This model provided a reasonable estimate for the market response, but may need to incorporate stochasticity in demand and user behaviour. Besides, solving the MPEC problem is another challenge due to the nature of MPEC problems.

Other than MPEC, approximate dynamic programming (ADP) has also being studied, which was able to solve this dynamic pricing parking challenge. ADP takes advantage of simulation techniques and parametric approximations to provide the estimate. Lei and Ouyang [LO17] brought up a non-myopic ADP approach along with the bi-level MPEC model to show the effectiveness of this method.

However, the approaches above are heuristics, which means the solution is case by case. The performance of a system might be affected by optimization error. Besides, solving the complexity of dynamic pricing problems, especially the dynamic programming part, is also a challenge. As a result, other approaches appeared.

3.2 Reinforcement learning

Reinforcement learning (RL) is one of the popular techniques, which takes advantage of the mathematical model with multiple variables. It is an agent-based AI algorithm that uses agents to optimize actions. In general, reinforcement learning can be formalized into a Markov Decision Process (MDP) as discussed in Section 2.2.1. Value function based approaches try to solve the MDP by finding the optimal value for a given state.

Q-learning is a value function based approach. Based on the given state and all successive steps, it can estimate the optimal action-value function from existing finite MDP.

Deep Q-learning [MKS+13] uses the deep convolutional neural network as a non-linear function approximator to represent the action-value function. It enhances the ability to support large amounts of actions.

Moreover, double deep Q-networks (double DQN) [vGS16] has been explicitly proposed focusing on reduced overestimation bias. For example, Rainbow [HMVH+18] combines the advantages of the following:

- distributional RL [SJLS00],
- multi-step learning [MKS+15],
- prioritized replay [SQAS16],
- double DQN (or DDQN) [vGS16],
- dueling DQN [WSH+16], and
- noisy net [FAP+18]

to provides enhanced performance. Policy search based approach is another strategy to solve an MDP. It aims at directly finding the optimal policies, usually by gradient-free or gradient-based methods.

Furthermore, the following used a trusted region to restrict the optimization steps:

- Trust Region Policy Optimization (TRPO) [SLA+15],
- Asynchronous Advantage Actor-Critic (A3C) [MBM+16],
- Proximal Policy Optimization (PPO) [SWD+17], and
- Soft Actor Critic (SAC) [HZAL18].

It can prevent policy updates being too extensive from previous policies.

Reinforcement learning is wildly used in areas such as nature language processing, motion control, recommendation system, etc. [GHLL17; LC17; SSS+17; ZZZ+18; KSRR19].

Demand response is another popular field that was involved with reinforcement learning. For example, Rana and Oliveira [RO15] found that different products may have interconnections with each other. Increasing the price or demand of product A may affect the demand of product B. Therefore, the researchers proposed a reinforcement learning Q-learning model to focus not only on the individual product, but also on multiple products. The result of this model showed its success on maximize the expected revenue of interdependent products and also optimal pricing of perishable interdependent products.

Kara and Dogan [KD18] connected reinforcement learning with an inventory management system of perishable products. It utilized Q-learning and Saras algorithm to balance the random demand of perishable products with deterministic lead time. In this system, a stock-based policy was used to replenish stock based on stock level, and an age-based policy was used to represent the inventory level and stocked item age.

Lu et al. [LHZ18] used reinforcement learning to study a smart grid system. The nature of the grid system indicated that people may have a high demand for electricity during day time, but have low demand during night. Also, industry and residency may take different priorities in demand. A reinforcement learning system was brought up based on the service provider's profit and customer's cost, to manage the relationship between supply and demand.

Mocanu et al. [MMN+19] built energy optimization based on deep Q-learning and deep policy gradient, to solved the same sequential decision problem and on-line scheduling of energy resources at the building level and the aggregated level.

3.3 Summary

In summary, *reinforcement learning* has shown its potential to the dynamic pricing system, and has been widely used on resource distribution and allocation systems such as the smart grid. On the other hand, the *dynamic pricing system* focuses on allocating the resource to urgent needs. Combining reinforcement learning along with dynamic pricing provide the ability to take care of the change of demands for limited resources.

Chapter 4

My dynamic pricing system

In this M.Sc. research on dynamic pricing systems, I am solving the following two questions: (a) As the available resource fluctuates year over the year, can the system adapt itself to the resource fluctuations? (b) As the system may take restrictions and these restrictions may change over time, can the model dynamically adapt to the constantly changing restrictions? To answer these questions, I propose in this chapter a system that consists of the following components: (a) a dynamic pricing system structure that utilizes artificial intelligence models, especially the machine learning-based time series prediction model; and (b) a reinforcement learning-based price control unit to address the questions above. The data set has been discussed in Chapter 5.

4.1 System overview

The main objective of this dynamic pricing system is aggregating multiple sources as the inputs then feed into the price control unit. One major source of inputs is the time-series based prediction. As a result, the dynamic pricing system comprises two components: the time series based prediction and reinforcement learning based dynamic pricing adjustment.

To elaborate on dynamic pricing, I introduce the content from time series prediction to provide a reference for the reinforcement learning model. It takes n inputs from historical data and produces n predictions. I use machine learning techniques, peculiarly the long short term memory model, to make the prediction.

As for the second part of the system, the price control unit will take into account the restrictions and inputs then provide the price adjustments. The core of the price control unit utilizes the advantages of reinforcement learning, especially deep Q learning, as the controller. It has the ability to find the optimal price based on the given equation. It will take several sources as inputs.

4.2 Time series prediction

Time series analysis and prediction have been developed for years. Several algorithms had been proposed, which have been discussed in Chapter 3. Here, I use the long short term memory (LSTM) as the main technology to process the time series data.

By comparing the traditional time series analysis techniques (e.g., ARIMA), LSTM

has several advantages which outperform ARIMA:

1. LSTM has been proofed LSTM with better performance on forecasting.

2. Due to the nature of recurrent neural network structure, it can take multiple parameters. As a result, it can be used for multivariate prediction.

Overall, I am using the time series analysis technique, especially the long short-term memory, to capture the temporal structure in time series data meanwhile make a prediction based on the temporal correlations.

4.2.1 LSTM model variants

I have three LSTM variants in my thesis—namely, the Encoder-Decoder LSTM, Stack LSTM and Vanilla LSTM:

1. The *Vanilla LSTM* is the simplest LSTM variant. It has three layers:

   (a) The first layer is an LSTM layer with 200 neurons.

   (b) The second layer is the dense layer with 100 neurons.

   (c) The last layer is a dense layer with the same number output as the period date.

   For example, if the forecasting period is 7 days, the output in the last layer should be 7. Both first and second layers use rectified linear unit (ReLU) as the activation function.

2. The *stack LSTM* variant has three layers:

(a-b) The first two layers are the LSTM layer with dropout. Both of them have 100 neurons. They also use the ReLU as the activation function.

(c) The last dense layer follows the number of period dates as the output.

3. The *encoder-decoder LSTM* variant has five layers:

   (a) It starts with a bi-directional LSTM with 200 neurons.

   (b) The second layer is the RepeatVector layer from TensorFlow. It adds an extra dimension to the data set.

   (c) The next layer is the bi-directional LSTM layer.

   (d) The fourth layer is the time distributed wrapped dense layer.

   (e) The last layer is another time distributed wrapped dense layer.

   The activation functions for encoder-decoders are ReLU.

4.2.2 Prediction period

To better discover the temporal correlations in the data set, I split the data based on time duration t. I define this time duration as a prediction period. For example, I will obtain a data set that contains 52 arrays if I process one year of data with a time duration has set to 7 (where $t = 7$). Each array in the new data set will contain seven elements. On the other hand, I will get a data set of 12×30 if I process the one-year data with t=30. It can be expressed as follows in the equation:

$$d(n/t, t) = f_{period}(n, t) \tag{4.1}$$

where

- n is the number of data in the data set, and
- t indicates the prediction period.

By utilizing the LSTM, I can make use of the algorithm to predict the trends based on different time duration. The prediction period can be varied based on the characteristic of the data set.

Example 4.2.1 Let us set the period date t to 7 days, which means the model will make a weekly prediction. Consider two time series capturing the first two weeks of bookings for parking. If there were 10 and 8 bookings for the first day of the first two weeks, then my algorithm predicts that there will be 9 bookings for the first day of the third week.

4.2.3 Weighted ensemble prediction

With the contributions of the prediction period, I can reveal the trends in different periods. The prediction from different periods will indicate the tendency based upon time interval. However, different time intervals will lead to extra flexibility, and robustness [AA12]. None of tendency that have flexibility and robustness can achieve uniformly best forecasts [AV16]. Also, since the prediction will be used for the price control unit, multiple predictions based on various time duration will be aggregated into a single but more precise prediction. It would benefit the down streams of the system.

The weighted ensemble is a common technique used in time series forecasting. It can combine forecasts from conceptually different methods, which enhance overall precision [AA12].

To enhance the prediction precision and merge the predictions, convex combination weighted ensemble technique has been used in the system. It can be expressed as follows:

$$\bar{x} = f_{ensemble}(w, x) = (w_1 \times x_1) + \ldots + (w_n \times x_n) \tag{4.2}$$

where

- w_i indicates the weight for i-th prediction.
- x_i indicates the i-th prediction value located from the prediction.
- n indicates the number of predictions.

Example 4.2.2 Consider two time series, in which (a) one capturing four weekly bookings for parking and (b) another one capturing a monthly booking for parking. With weights w_1=0.7 and w_2=0.3, if there were x_1=10 bookings on the first day of the first week and x_2=14 bookings on the corresponding first day of the month, then my algorithm predicts that there will be $\bar{x} = 0.7 \times 10 + 0.3 \times 14 = 11.2$ bookings on that day.

4.2.4 Algorithm for time series prediction

The time series prediction can be expressed as Algorithm 1. The algorithm takes normalized time series data as the input. The first step from lines 1-3 utilizes Equa-

tion (4.1) to transfer the normalized data into time series entries based on the prediction period. The actual prediction is made in the second step from lines 4-17. Each prediction period data set are used to make prediction and saved for the next step. In the second step, I first split the data into training and test data sets. I use 80% of the data as the training data set and the remaining 20% as the test data set. Then, the algorithm creates and trains the LSTM model with the training data set in line 8. Also, it accepts different parameters that affect LSTM model accuracy. The next step after the model has been trained is walking forward prediction which if from lines 9-17. It takes steps from the test data set and makes predictions. Each prediction it made should equal to the length prediction period. For example, the prediction should be 7 if the prediction period set to 7. At the end of the walking forward prediction, predictions based on different prediction periods will be saved. The last step of the prediction algorithm is to ensemble the predictions (lines 18-21). In more details, I generate an array with a preset step n. The elements in the array fall into the range of [0, 1]. For example, the number will be [0, 0.01, 0.02, 0.03, ..., 1]. Then, I use brute force to generate the convex combinations from previous array based on the condition that the sum of the convex combination should always less than or equal to 1, i.e., [0, 1]. Next step, Equation (4.2) makes use of a series of convex weight to merge multiple predictions into one final prediction. Example 4.2.2 reveals the detail of this step. The convex combinations that provides the best performance will be used as the selected combination for future use.

Algorithm 1: Time Series Prediction

Data : Normalized Time Series Data Set, Prediction period

Result: Merged Prediction

for $t \in$ *prediction period* **do**
    period data sets $\leftarrow f_{period}(n, t)$
end
prediction $\leftarrow \emptyset$
for *data* $\in$ *period data sets* **do**
    train, test $\leftarrow$ *data*
    # Train the model
    model $\leftarrow f_{fit_{LSTM}}(train, params)$
    # Walking forward prediction
    history $\leftarrow \emptyset$
    for *entry* $\in$ *test* **do**
        history.append(entry)
        yhat $\leftarrow f_{forecast}(model, history)$
        prediction.append(yhat)
    end
    predictions.append(prediction)
end
Ensemble predictions
Ensembled prediction $\leftarrow f_{ensemble}(predictions, weights)$
MAPE $\leftarrow$ mean_absolute_percentage_error(Ensembled prediction, test)
return (Ensembled prediction, MAPE)

4.3 Reinforcement learning for dynamic pricing

In studying dynamic pricing, determining the best price for customers based on the estimated demand while under policy restriction is the challenge. In this section, I solve the following questions:

1. How can we distribute the limited resource to the people that urgently need the resource?

2. How to integrate different policy restrictions into the model (e.g., what if the policy regulates the final price should be less than a certain amount)?

In this section, I propose the price control unit. The price control unit is the primary function of this application. It utilizes the reinforcement learning model, especially Q-learning. More specifically, I convert the problem above into a Markov Decision Process (MDP) with a tuple $\langle \mathcal{S}, \beta, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$.

In my thesis, the price control unit will take demand estimation as input on behalf of the state change. Then, the price control unit uses the Q value function to evaluate the state change and provide the state-action value (Q-value). The optimal Q value function should be able to provide the maximum return.

4.3.1 State space

The state's space contains all the possible states of the environment. In this thesis, the state will be the parking price. As the result, we define state space $\mathcal{S}$ as follows:

$$\mathcal{S} = s \tag{4.3}$$

where $s \in \mathbb{Q}^+$.

4.3.2 Action space

The action space demonstrates all the possible actions that this model can take. As I apply this technique to the parking management field, the possible action in this situation to limit to three types of action—namely, price increase, price drop, price hold:

- Price hold will hold the current price without change.
- In contrast, price increase and price drop will update the current price to a higher or lower price.

Here, I introduce two variables to define the available actions better:

1. The minimum price change amount m.
2. The price change step n.

The price adjustment amount falls into the range of the magnification of the minimum price change with the price change step, i.e., $[-m \times n, +m \times n]$. For example, if we set the m=\$1 within n=5 steps, then the price can be increased from [+\$1, +\$5] while the price drop will be in the range of $[-\$5, -\$1]$.

In this thesis, we used the action space $\mathcal{A}$ as follows:

$$\mathcal{A} = n * m \tag{4.4}$$

where $m, n \in \mathbb{Q}$. Moreover, the summation of $n * m$ with current price p should in the range of price restriction.

$$\mathcal{R}_{lower} \leq n * m + p \leq \mathcal{R}_{upper} \tag{4.5}$$

where

- $\mathcal{R}$ indicates the price restriction
- p indicates the current price

4.3.3 Reward space

Reward function is one of the most important parts in dynamic pricing model. It can quantified multiple given variables by given formula. Also, it provides the ability to evaluate how good is the state used for enhancing the model.

Since the reward function is the core function in the system, the equation is able to adapt the actual use case. To elaborate, mathematically, consider the following parameters:

- adjustment ceiling (maximum) ϵ_m
- estimate demand duration δ
- time slot t
- type of parking lot v
- total parking resource n

- total revenue R
- parking resource threshold h
- available parking resource θ

Then, the reward function fulfills the follows:

1. Take the prediction of the estimated demand duration δ_t at time slot t.
2. Provide the price adjustment amount $\epsilon_{t,v}$ for time t at type of parking lot v , under a preset parameter ϵ_m which indicates the price ceiling.
3. Maximize the revenue R based on demand and cost, taking into consideration of time range [t,T] with number of parking resource [n, N] and types of parking lot [v, V].

Moreover, we also incorporate the constraints:

1. estimated demand duration δ should be within the available resource θ,
2. available resource should be at most the difference between total resource n and the preset parameter threshold h (i.e., $n - h$), and
3. cost parameter $\epsilon_{t,v}$ should be at most the preset parameter cost ceiling ϵ_m.

More formally, to express the problem in a formula:

$$R = \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{N} \epsilon_{t,v} \cdot \delta_t \tag{4.6}$$

$$\text{such that } 0 \leq \delta_{t,c} \leq \theta = n - h \tag{4.7}$$

$$0 \leq \epsilon_{t,v} \leq \epsilon_m \tag{4.8}$$

where

- $\epsilon_{t,v}$ indicates the daily parking rate, and
- δ_t indicates the duration of the customer parking.
- $\delta_{t,v}$ indicates the duration of the customer parking for a specific parking facility.

Combine them together, I get the earnings for this parking event $\epsilon_{t,v} \cdot \delta_t$. By given the reward function, I get the total earnings:

$$\sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{N} (\epsilon_{t,v} \cdot \delta_t) \tag{4.9}$$

based on the accumulation individual earnings with types of parking lot V, the duration of parking T, and the number of parking lot used N.

To define the reward function, one may attempt to apply the discrete finite Markov Decision Process (MDP) based on the demand function created in the previous step. The MDP contains a tuple $\langle \mathcal{S}, \beta, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ with reward function (where $\beta \in \mathbb{Z}^+$ indicates the sets of available parking price):

$$\mathcal{R}(s, a, s') = \begin{cases} 1 & \text{if} \quad R_s > 0 \\ -1 & \text{if} \quad R_s < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where

- $s \in S$,
- $a \in \mathcal{A}$, and

- $s' \in S$.

The goal of using reinforcement learning is to learn to choose a sequence of actions that can obtain optimal actions to gain maximum rewards. However, the reward in Equation (4.10) is not accurate enough to reveal the changes on earning. For example, the \$10 and \$200 extra earning shows the same reward.

To enhance it, I define the reward function as follows. Since the earnings price already show the numerical difference, I can simply use the earnings as the reward. The resulting reward function would be:

$$\mathcal{R}(s, a, s') = \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{N} (\epsilon_{t,v} \cdot \delta_t) \tag{4.11}$$

where

- $s \in S$,
- $a \in \mathcal{A}$, and
- $s' \in S$.
- $\epsilon_{t,v}$ indicates the daily parking rate, and
- δ_t indicates the duration of the customer parking.

4.3.4 Algorithm for price control

Here, I demonstrate the logic of the pseudo-code which refers to Algorithm 2. For each epoch, the algorithm has to reset the state and adjustment and initiate a time series data set that shows in lines 1-4. Since the price control unit does not have

a clear termination condition, it stops at the end of time series data. Hence, the for-loop in line 5 stops at the end of time series data.

Lines 6-8 stand for the actual logic to update the model and make a price adjustment. It picks an action from the state-action pair (Q-values). Then, it evaluates the action-based picked action and the prediction from time series. The reward function provides the reward and next state-action. A numerical example of lines 6-8 is shown in Example 4.3.1.

Example 4.3.1 My dynamic pricing model uses the state information retrieved from the previous state, especially the daily parking price (i.e., $p = \$12$). Meanwhile, the model reads the predicted number of daily bookings (i.e., $ts = 18$ bookings). Then, the model calculates both the reward and the next state based on p and ts. By doing so, the model provides the price adjustment $adj = \$4$. The consequent daily parking price will become $adj + p = \$12 + \$4 = \$16$.

Algorithm 2: Price Control Unit

Data : Time Series Prediction Data Set

Result: Price adjustment

```
for e ∈ epoch do
    state ← Environments
    ts ← Time Series Data Set
    adjustment ← ∅
    for i = 1, size of time series do
        action ← Q(state[i])
        reward, state′ ← R(action, ts[i])
        adjustment ← state′
    end
end
```

4.4 Summary

To summary, my proposed dynamic pricing system utilizes artificial intelligence models. These include (a) the machine learning-based time series prediction model and (b) reinforcement learning-based price control unit. In more details, the prediction on the number of bookings for parking was made based on three types of LSTM variants on different periods. Ensemble techniques were merged to the time series data to aim for optimal predictions on the number of bookings.

As for the price adjustment model, I defined all the action space, state space and reward space. Here, the action space captures three types of actions: price increase,

price hold, and price decrease. The state space captures numerical value. Finally, I designed the reward function based on the requirement or constraints. The results predict the price adjustments, and thus their impacts on the extra profile (or loss).

Chapter 5

Evaluation

The experiments have been split into three parts. First, I introduce the data set and experiment environment. Then, I bring in the experiments for time series analysis. In the end, I introduce the outcomes from the overall model.

5.1 Experiment setup

5.1.1 Data set

The primary data set I used is a real-life parking data set from a mid-sized Canadian airport. The data contain 25,144 rows of records capturing parking information from January 15, 2015 to February 29, 2020 inclusive, for a total of 1,884 days.

The data set contains three types of parking lots which can be treated as three separate data sets. They are

1. parkade (PKD),

2. economy parking lot (ECO), and

3. valet parking (VAA).

Each type of parking lot has its charging standards. Each record from the data set stands for an individual parking booking from the customer. The record contains booking time, arrival time and booking duration. In this experiment, I mainly use arrival time for a prediction.

5.1.2 Data pre-processing

I applied few data prepossessing techniques to massage the data. I utilized min-max techniques to normalize the data. Min-max is mainly used for feature scaling. The main feature of min-max is to map the values from different scales under a standard scale. It benefits algorithms that use the Euclidean distance from unified scales. It follows the the following equations:

$$X_{scaled} = \frac{x' - V_{min}}{V_{max} - V_{min}} \tag{5.1}$$

where

- x' represents each value that need to be processed,
- V_{max} is the maximum of the given range, and
- V_{min} is the minimum of the given range.

In addition, I extracted the number of user arrivals per day from the booking data set since the prediction is on a daily basis. By the end of this process, I have collected

three data sets indicating each parking lot type. Each data set contains 1884 rows of data representing the number of booking for this data set for that day. To follow the best practice for prediction, I split the data set into three portions:

- 64% data for training,
- 16% data for validating, and
- 20% of the data for testing.

5.1.3 Experiment environment

The program is mainly implemented in Python with TensorFlow 2.0 [AAB+15] and scikit-learn library [PVG+11]. The research utilizes the AWS P2 xlarge instanc [Ama21] as the experiment machine. AWS P2 xlarge instance has one NVIDIA K80 GPU, four vCPU cores and 61 Gib memory. The operation system on the AWS P2 xlarge instance is Ubuntu 18.04 LTS.

5.2 Time series analysis

5.2.1 Parameters

The LSTM variants have multiple hyperparameters, such as the basic time steps and batch size. Since we have three data sets, the best hyperparameter may vary from data set to the data set. As a result, I pick the best parameters in range as shown in Table 5.1. In addition, I also automated the hyperparameter optimization process by using the grid search [LBL04] method.

Table 5.1: Parameters for the LSTM

Param Name	Content
Batch size	32, 128, 512
Optimizer	Adam, Nadam
Epochs	500, 1000, 1700
Dropout rate	0.1, 0.2
Loss	MAPE, RMSE

Other than the parameters listed from Table 5.1, I follow the recommended parameters from TensorFlow for the optimizers listed in Table 5.2 for the Adaptive Moment Estimation (Adam) variants.

Table 5.2: Adam optimizer variants

Optimizer Type	α	β_1	β_2	ϵ
Adam	0.001	0.9	0.999	1.00E-07
Adamax	0.001	0.9	0.999	1.00E-07

5.2.2 Evaluation metrics

To better evaluate the outcomes, especially the accuracy of the forecast, I applied multiple evaluation metrics:

- mean absolute error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| \tag{5.2}$$

  indicates the measures from predictions with actual data. Here, (a) f_i stands for the forecast value and (b) y_i stands for the actual value.

- Root Mean Square Error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - f_i)^2}{n}} \tag{5.3}$$

  which indicates the standard deviation of the prediction errors.

- mean absolute percentage error (MAPE):

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{y_t - f_t}{y_t} \right| \tag{5.4}$$

  which is a percentage error for revealing how accurate a forecast system is.

Low values for MAPE, MAE and RMSE indicates high accuracy.

5.2.3 Results and discussions

To better understand the prediction result for the LSTM time series forecasting, I also introduced the Autoregressive Integrated Moving Average Model (AIMAM) as the baseline. I also follow the suggested parameters ARIMA model. Since I have applied the ensemble techniques, the ARIMA predictions also got merged with a convex factor into an individual forecasting.

The experiment has employed multiple periods for individual forecasting. In more details, I used 7, 30, 60, 120 days for forecasting. The seven-day prediction from Table 5.3 shows that Encoder-Decoder achieves outstanding results in VAA and PKD data sets. Encoder-Decoder had at least 23.5% error less than the baseline. Stack LSTM also gets the best results in the ECO data set. The 30-day prediction from

Table 5.4 shows that ARIMA outperforms other models on the ECO and PKD data set. Encoder-Decoder still achieves the best results in the VAA data set. The 60-day prediction from Table 5.5 shows that Vanilla achieves the best results in VAA and PKD data set while Encoder-Decoder decreased at least 22.4% error than the Vanilla in ECO data set. The 120-day prediction from Table 5.6 shows that Encoder-Decoder achieves outstanding results in VAA and PKD data sets. Encoder-Decoder had at least 23.5% less error than the baseline. Stack LSTM also gets the best results in the ECO data set.

By comparing the outcomes from periods refers to Tables 5.3, 5.4, 5.5 and 5.6, I observed that the 30 days periods has the largest error among all four tables. Besides, the seven days predictions have the lowest forecasting error, which has the best prediction result. Other than that, the baseline ARIMA shows the ability to predict on 30 days data where all other LSTM variants shows a lousy performance. However, since 30 days has the most significant error among all predictions, that may mean 30-day prediction may not represent the nature of the human activity. It may not be a good prediction option.

Among all three LSTM variants, the encoder-decoder provides the best predictions. Surprisingly the Vanilla LSTM has a better result than Stack LSTM in most cases. This may because of overfiting in the training process. Adding a dropout layer may change the situation.

Other than that, I use the MAPE and RMSE as the loss function while using the MAE, MAPE and RMSE as the evaluation metric. However, the outcomes indicate that most predictions achieve minor errors in metrics with MAPE as the loss function.

As a result, MAPE has been chosen as the major evaluation metric.

Table 5.3: Evaluation of time series analysis with period = 7 days

Model	dataset	period	RMSE	MAPE	MAE
ARIMA (baseline)	VAA	7	0.089	48.946	0.058
My Encoder Decoder LSTM	VAA	7	0.102	**38.690**	0.063
My Stack LSTM	VAA	7	0.151	145.591	0.114
My Vanilla LSTM	VAA	7	0.087	41.089	0.058
ARIMA	ECO	7	0.131	26.503	0.101
Encoder Decoder	ECO	7	0.135	27.823	0.103
Stack	ECO	7	0.126	**26.003**	0.096
Vanilla	ECO	7	0.128	26.454	0.098
ARIMA	PKD	7	0.193	37.018	0.108
Encoder Decoder	PKD	7	0.153	**28.311**	0.084
Stack	PKD	7	0.154	37.816	0.094
Vanilla	PKD	7	0.187	30.820	0.098

Table 5.4: Evaluation of time series analysis with period = 30 days

Model	dataset	period	RMSE	MAPE	MAE
ARIMA	VAA	30	0.158	98.061	0.118
Encoder Decoder	VAA	30	0.180	**88.445**	0.143
Stack	VAA	30	0.181	89.368	0.144
Vanilla	VAA	30	0.205	196.667	0.151
ARIMA	ECO	30	0.195	**43.259**	0.150
Encoder Decoder	ECO	30	0.212	44.075	0.160
Stack	ECO	30	0.212	43.474	0.159
Vanilla	ECO	30	0.241	54.248	0.182
ARIMA	PKD	30	0.199	**73.432**	0.139
Encoder Decoder	PKD	30	0.337	185.548	0.256
Stack	PKD	30	0.289	79.406	0.189
Vanilla	PKD	30	0.312	120.443	0.221

Table 5.5: Evaluation of time series analysis with period = 60 days

Model	dataset	period	RMSE	MAPE	MAE
ARIMA	VAA	60	0.187	113.239	0.129
Encoder Decoder	VAA	60	0.181	84.126	0.129
Stack	VAA	60	0.204	196.999	0.151
Vanilla	VAA	60	0.143	**75.712**	0.116
ARIMA	ECO	60	0.288	70.764	0.235
Encoder Decoder	ECO	60	0.221	**46.350**	0.167
Stack	ECO	60	0.287	75.609	0.231
Vanilla	ECO	60	0.253	59.795	0.199
ARIMA	PKD	60	0.302	94.777	0.214
Encoder Decoder	PKD	60	0.345	200.000	0.266
Stack	PKD	60	0.233	65.842	0.176
Vanilla	PKD	60	0.264	**60.152**	0.165

Table 5.6: Evaluation of time series analysis with period = 120 days

Model	dataset	period	RMSE	MAPE	MAE
ARIMA	VAA	120	0.185	112.181	0.155
Encoder Decoder	VAA	120	0.152	83.669	0.137
Stack	VAA	120	0.153	**67.554**	0.098
Vanilla	VAA	120	0.205	198.957	0.151
ARIMA	ECO	120	0.148	**30.361**	0.113
Encoder Decoder	ECO	120	0.281	77.632	0.236
Stack	ECO	120	0.282	76.503	0.234
Vanilla	ECO	120	0.255	62.367	0.205
ARIMA	PKD	120	0.212	78.335	0.167
Encoder Decoder	PKD	120	0.277	**67.347**	0.174
Stack	PKD	120	0.344	198.892	0.266
Vanilla	PKD	120	0.273	67.767	0.173

5.2.4 Ensemble results

The overall ensemble forecasting results for each data set are as below:

Table 5.7: Ensemble predictions

Data set	Model	RMSE	MAPE	MAE
VAA	ARIMA (baseline)	0.087	50.731	0.056
VAA	LSTM	**0.072**	**36.346**	**0.050**
ECO	ARIMA	0.129	29.117	0.099
ECO	LSTM	**0.126**	**26.003**	**0.096**
PKD	ARIMA	0.186	40.923	0.102
PKD	LSTM	**0.153**	**28.311**	**0.084**

From Table 5.7, I obtained that LSTM led to 10.7% lower error than ARIMA in MAPE for ECO. Meanwhile, PKD led to 34.87% lower, and VAA led to 14.5% lower in MAPE error. By comparing the ensemble outcomes with individual predictions, the ensemble result has shown a 6% decrease in MAPE error in the VAA data set. The ensemble techniques work as expected.

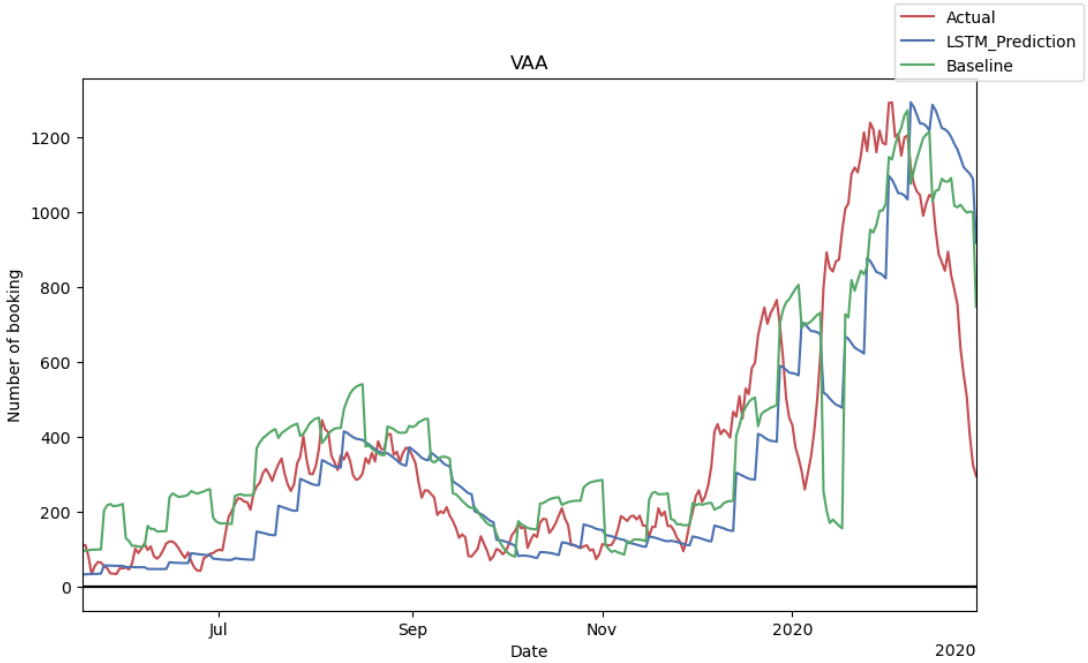

Figure 5.1: Results of LSTM prediction on the number of bookings for the VAA data set

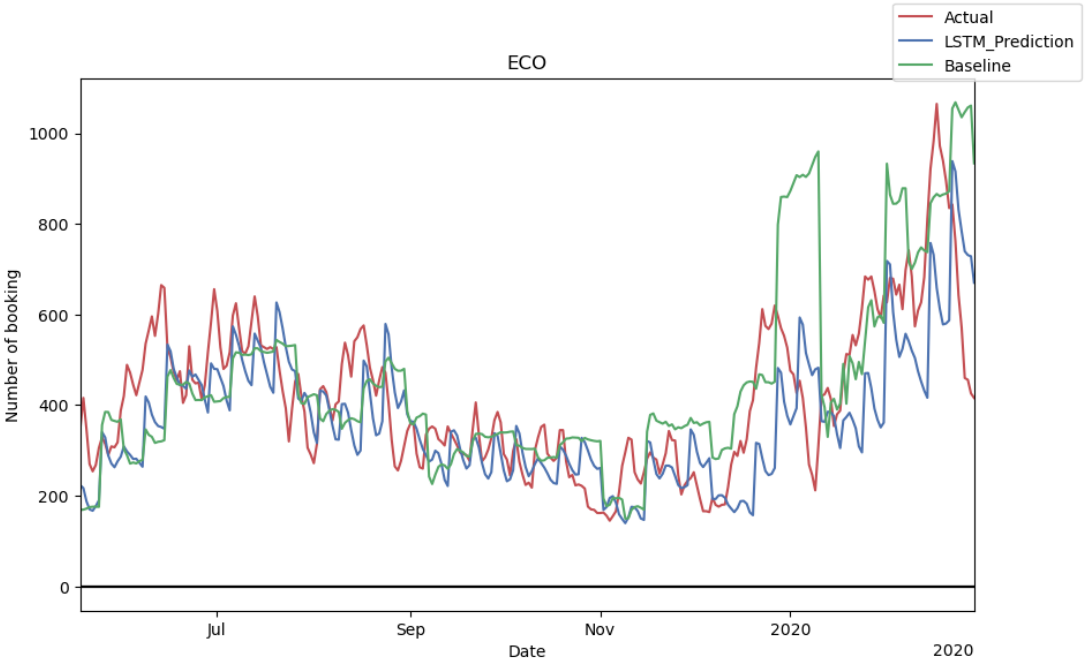

Figure 5.2: Results of LSTM prediction on the number of bookings for the ECO data set

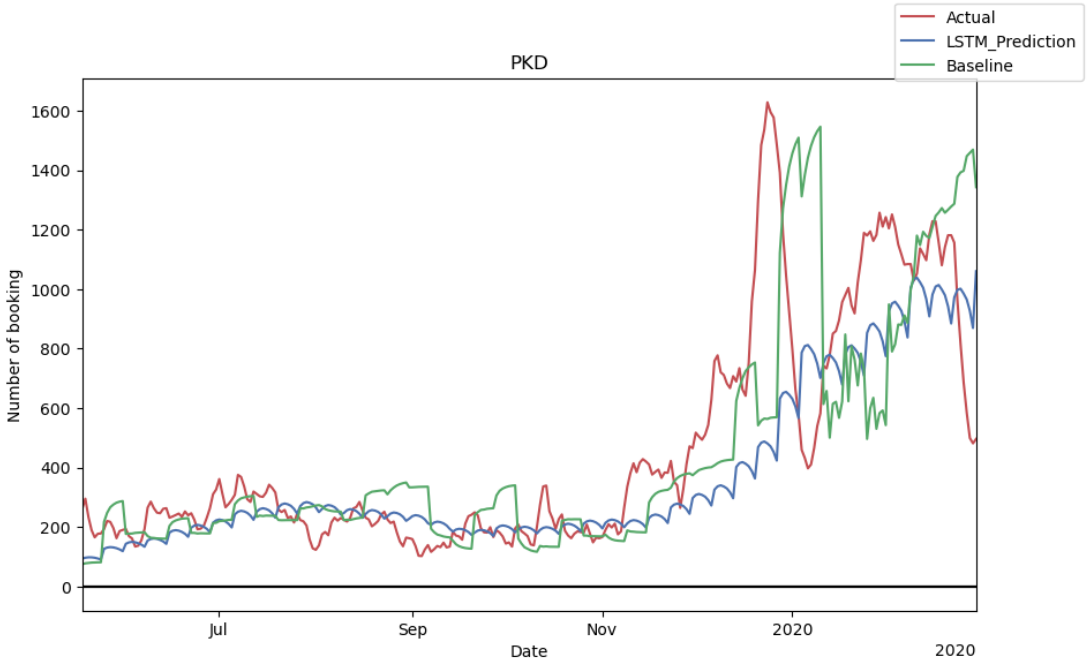

Figure 5.3: Results of LSTM prediction on the number of bookings for the PKD data set

Figures 5.1, 5.2 and 5.3 show the comparison of LSTM ensemble predictions, ARIMA ensemble predictions and the actual data plotted in testing data sets. From the plotting, I observed that LSTM predictions for all three data set follow a similar pattern as actual data. In more details, LSTM results in Figure 5.1 for VAA data set has missed the peak at 2019 Christmas because holiday does not count as an impact factor for bookings. The data from all three types of data sets show a severe decrease because of the COVID-19 situation.

To summarize, the forecasting from LSTM outperforms the ARIMA in all three data sets.

5.3 Dynamic pricing

5.3.1 Model selection

It used the Double Deep Q network [vGS16]. There are four dense layers that exist. The first dense layer has 24 neurons. The second dense layer has 48 neurons, while the third has 96 neurons. All of them use ReLU as the activation function. The last dense layer has three neurons along with a linear activation function.

Regarding the model parameters, I set the learning rate to 0.001, batch size at 64, I chose Adam as the model optimizer. I followed the suggested parameters for the optimizer as showing in Table 5.1.

5.3.2 Parameters

There are multiple parameters have been set for this experiment. Expressly, I have set the price adjustment range from \$0 to \$7, i.e., [\$0, \$7]. It follows the same test data set as time-series predictions which starts from May 2019 till Feb 2020.

The episode has been set to 2000. The discount factor γ has been set to 0.96. It will be used in the Bellmen equation (Equation (2.6)) to find the optimal policy.

5.3.3 Evaluation

Since the reinforcement learning on the price adjustment does not have a direct evaluation metric, here I have to find a different approach to evaluate the outcome for the application. Consider the purpose of the adjustment unit is to adjust the price and make a profit, I evaluate the price adjustment unit by checking the earnings.

5.3.4 Results

Table 5.8: Price adjustment unit earnings

Month	VAA($)	ECO($)	PKD($)
2019-May	532.60	2,173.30	1,169.90
2019-Jun	4,249.20	18,725.15	904.25
2019-Jul	6,565.10	20,693.95	3,500.40
2019-Aug	12,354.10	12,911.50	16,404.45
2019-Sep	5,194.80	16,406.75	29,865.65
2019-Oct	2,074.45	3,666.20	13,529.90
2019-Nov	9,620.70	1,860.20	13,210.00
2019-Dec	8,197.00	2,198.35	11,155.80
2020-Jan	36,894.15	9,698.00	29,067.95
2020-Feb	22,601.40	17,930.45	59,209.60
SUM:	108,283.50	106,263.85	178,017.90

Earnings unit in Canadian dollar ($)

I have grouped the outcomes for this component in Table 5.8. From the sum of the table, I find out the overall results meet the expectations.

In more details, the earnings for VAA were relatively high in 2019 August, November, December and January 2020. The earnings for ECO data set relatively high in 2019 July, September and 2020 January and February. The earnings for PKD data set relatively high in 2019 September, 2020 January and February.

By comparing the high earning period with the actual data for VAA and PKD, the 2020 January has higher earnings than 2019 December. The main reason is that bookings start increasing in the second half of December due to the Christmas holiday. Meanwhile, people prefer taking flights in the whole of January and February due to the cold weather.

ECO data set represents the economic parking lot. From the ECO data set result, I observed that people have high demands in summer time (June and July), September,

and February 2020. Summer and September are the best time in the year for travel, while February is the spring break in schools. This may indicate economic parking lot parking users will choose the best time for travel.

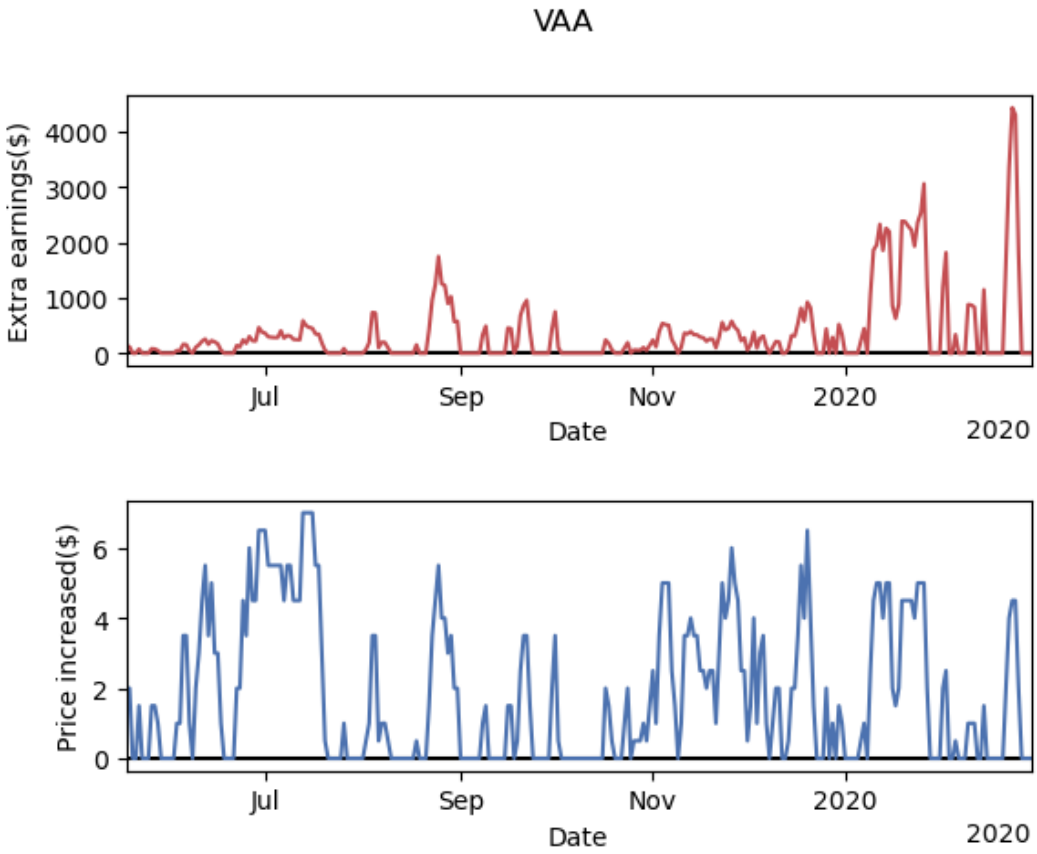

Figure 5.4: Results of double deep Q network prediction on price adjustment for the VAA data set

Price in Canadian dollar ($)

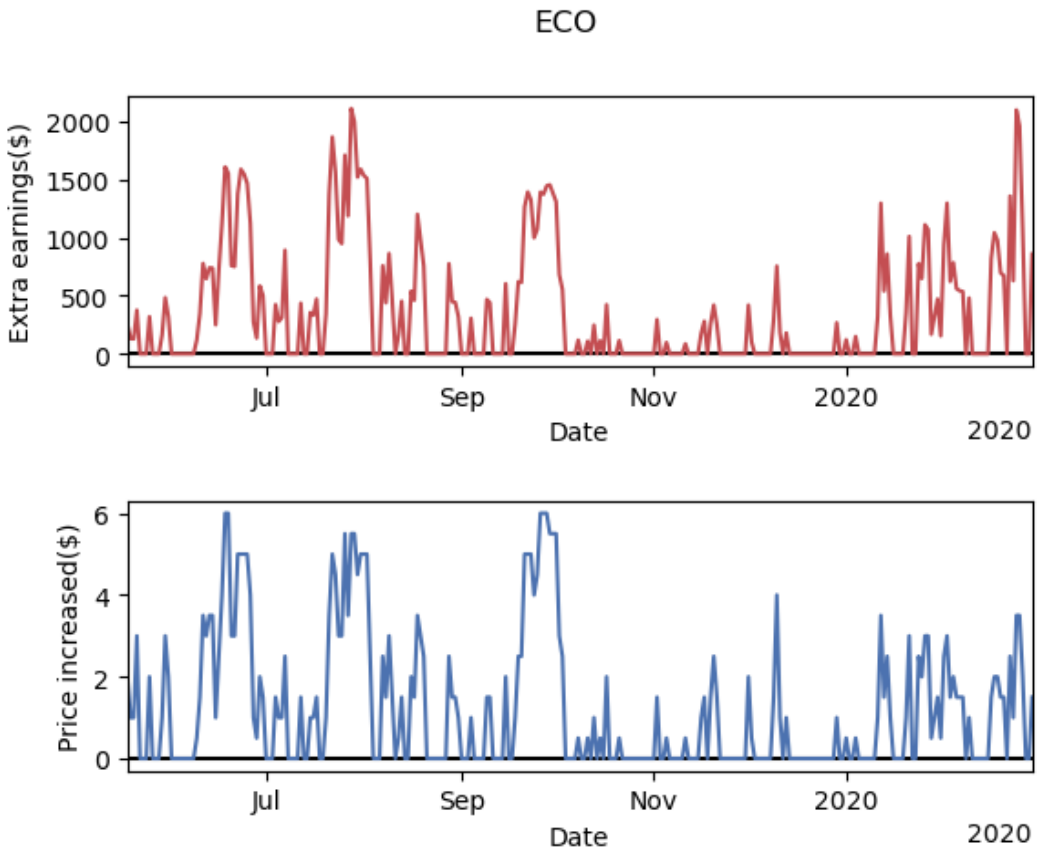

Figure 5.5: Results of double deep Q network prediction on price adjustment for the ECO data set

Price in Canadian dollar ($)

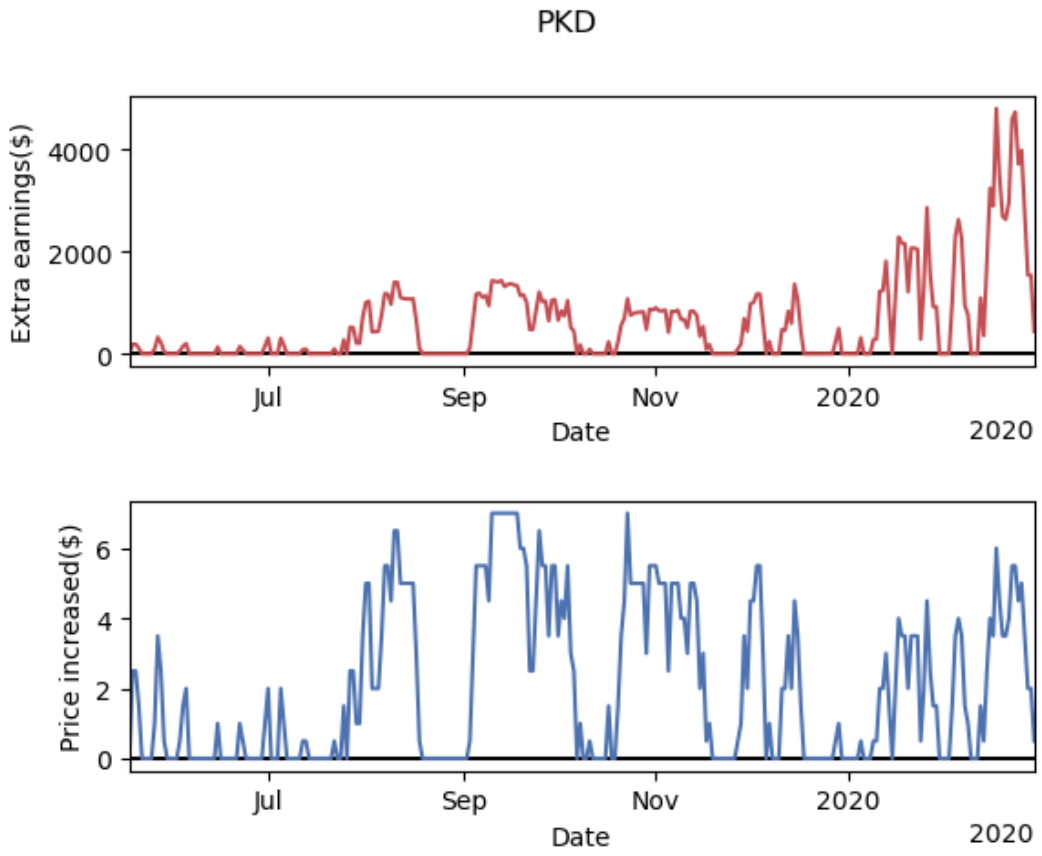

Figure 5.6: Results of double deep Q network prediction on price adjustment for the PKD data set

Price in Canadian dollar ($)

Besides, by comparing the earnings with price adjustment by the timeline in Figures 5.5, 5.4 and 5.6, I observe that price adjustment will effectively bring the earnings. For example, the price increment from $0 to $7 brings in earnings over $2000 per day in 2020 February in PKD data set. However, the price adjustment from $0 to $7 for the VAA data set in July 2019 only brings in less than $1000 earnings. The reason behind this is that the algorithm may be too sensitive to fluctuations. This can be an enhancement work in the future.

To summarize, the price adjustment unit can adjust the price based on predictions

and bring in more earnings under restrictions.

5.4 Summary

In summary, I evaluated our models. The booking estimate unit utilizes the time series prediction to make the forecasting. Specifically, it utilized three LSTM variants for the predictions with different periods. Then, I used convex ensemble techniques to merge the predictions into one prediction that outperformed other individual predictions. To better evaluate the quality of the LSTM, ARIMA was introduced as the baseline. Performance was measured by the MAPE as the evaluation metric. Evaluation results indicated that the LSTM prediction outperformed the ARIMA on all three parking lots. Specifically, the ensemble technique for ECO and PKD data sets follows the same prediction as the 7 days prediction. However, the VAA data showing ensemble technique had 6% less error than the 7 days prediction in MAPE error. As for the price adjustment unit, I used profit as the evaluation metrics. The monthly outcomes show that, as expected, all of the three parking lots were making profits. Moreover, the data also showed that ECO parking lot users preferred to use the parking lot during the holiday season (e.g., July, Spring break) while PKD and VAA earned more in August and January.

Chapter 6

Conclusions and future work

6.1 Conclusions

In my thesis, I proposed an architecture that utilizes the available public resource while optimizing revenue with pre-defined restrictions, specifically on the parking management field. The architecture has two components: the time series based booking estimate unit and the reinforcement learning based price control unit. I used a real-life parking data from a mid-sized Canadian airport. It consists of 25,144 rows of records capturing three types of parking from January 15, 2015 to February 29, 2020, for a total of 1,884 days. In more details, I designed an LSTM based neural network used for time series prediction as to the booking estimate unit. Evaluation results show that LSTM based neural network outperforms the traditional ARIMA model in all three types of parking lot data. Besides, I designed another dynamic pricing model based on reinforcement learning. I combined and adapted the booking estimate unit with the price control unit together with a preset price restriction. I

chose the overall profit as the evaluation metrics for the architecture. The outcome of the overall profit met the expectations.

Recall that, in Chapter 1, we asked several questions. I provided answers to these questions in this thesis. Let me summarize them:

Q1. As the available resource fluctuates year over the year, can the system adapt itself to the resource fluctuations?

As described in Section 4.2, although bookings may fluctuate by various factors year over the year, the time series prediction—especially the LSTM prediction model—adapts to the data fluctuation.

Q2. How can we distribute the limited resource to the people that urgently need the resource?

As described in Section 4.3, the limited resource can be assigned to the people that urgently need by the price.

Q3. How to integrate different policy restrictions into the model (e.g., what if the policy regulates the final price should be less than a certain amount)?

As described in Section 4.3 (in particular, Section 4.3.3), the price will be controlled by the price control unit made by deep reinforcement learning. The price control unit is able to adjust the price based on multiple inputs. This thesis uses the prediction data provided from the booking estimate unit, which utilizes the time series prediction. Moreover, the deep reinforcement learning model can take restrictions that conform to the real world case that the policy may regulate parking lots.

Q4. After concatenate different components, what would be the best metrics to evaluate the overall consequence?

As described in Chapter 5, I presented the overall earnings as an evaluation metrics to measure the overall consequence. From the experiment, the earnings from price adjustment significantly increase the earnings.

6.2 Future work

In this thesis, I proposed an architecture that utilizes the available public resource while optimizing revenue with predefined restrictions, especially in the parking management field. As future work, one could explore the following:

1. With the development of the time series prediction, the prediction model can be replaced by a more accurate model. In more details, with the development of the time series prediction, more popular and influential prediction techniques may appear. For example, the LSTM variants could be extended with GRU or attention mechanism.

2. Other than the uni-variant time series prediction, more data sets can be introduced, such as the statutory holiday's data set. The LSTM had shown its power on multivariate prediction. By combining multiple data set, the prediction model may provide a more accurate prediction.

3. The dynamic pricing system reveals the potential to take more input. For example, the flight data may be highly correlated with the parking data. More input may help the system to adjust the price better based on the needs.

4. Currently, the model only supports the daily adjustment per request. However, since the raw data set records are logged with date and time. An hourly adjustment system could be developed by utilizing the data set.

5. To better handle the price restriction with negative numbers such as [−$3, $4], the supply and demand curve may help to provide a better reference to drop the price. Other than that, it will explain that why drop the price for a certain amount. For example, it can explain why to drop $1 instead of $3.

Bibliography

[AA12] Ratnadip Adhikari and R. K. Agrawal. A novel weighted ensemble technique for time series forecasting. In Pang-Ning Tan, Sanjay Chawla, Chin Kuan Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, pages 38–49, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Ama21] Amazon Web Services. Amazon ec2 p2 instances, 2021.

[AS18] P. Arumugam and R. Saranya. Outlier detection and missing value in seasonal arima model using rainfall data*. *Materials Today: Proceedings*, 5(1, Part 1):1791–1799, 2018. International Conference on Processing of Materials, Minerals and Energy (July 29th – 30th) 2016, Ongole, Andhra Pradesh, India.

[AV16] Ratnadip Adhikari and Ghanshyam Verma. Time series forecasting through a dynamic weighted ensemble approach. In Atulya Nagar, Durga Prasad Mohapatra, and Nabendu Chaki, editors, *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*, pages 455–465, New Delhi, 2016. Springer India.

[BGV+20] Domenico Benvenuto, Marta Giovanetti, Lazzaro Vassallo, Silvia Angeletti, and Massimo Ciccozzi. Application of the arima model on the covid-2019 epidemic dataset. *Data in Brief*, 29:105340, 2020.

[BJ90] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., USA, 1990.

[Den15] Arnoud V. Den Boer. Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science*, 20(1):1–18, June 2015.

[DYMJ19] Yuchuan Du, Shanchuan Yu, Qiang Meng, and Shengchuan Jiang. Allocation of street parking facilities in a capacitated network with equilib-

rium constraints on drivers' traveling and cruising for parking. *Transportation Research Part C: Emerging Technologies*, 101:181–207, April 2019.

[FAP+18] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[GHLL17] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[Ham95] James D Hamilton. Time series analysis, 1995.

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning

with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[HZL+21] Zhongyang Han, Jun Zhao, Henry Leung, King Fai Ma, and Wei Wang. A review of deep learning models for time series prediction. *IEEE Sensors Journal*, 21(6):7833–7848, 2021.

[IL15] Eren Inci and Robin Lindsey. Garage and curbside parking competition with search congestion. *Regional Science and Urban Economics*, 54:49–59, September 2015.

[JWW+18] Ruili Jiang, Haocong Wang, Han Wang, Eoin O'Connell, and Sean McGrath. Smart parking system using image processing and artificial intelligence. In *2018 12th International Conference on Sensing Technology (ICST)*, pages 232–235, 2018.

[KD18] Ahmet Kara and Ibrahim Dogan. Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. *Expert Systems with Applications*, 91:150–158, 2018.

[KSRR19] Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. Deep Reinforcement Learning for Sequence-to-Sequence Models. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, aug 2019.

[LBL04] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. On the relationship between classical grid search and probabilistic

roadmaps. *The International Journal of Robotics Research*, 23(7-8):673–692, 2004.

[LC17] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[LHZ18] Renzhi Lu, Seung Ho Hong, and Xiongfeng Zhang. A Dynamic pricing demand response algorithm for smart grid: Reinforcement learning approach. *Applied Energy*, pages 220–230, June 2018.

[LHZS16] Chenghao Liu, Steven CH Hoi, Peilin Zhao, and Jianling Sun. Online arima algorithms for time series prediction. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[LO17] Chao Lei and Yanfeng Ouyang. Dynamic pricing and reservation for intelligent urban parking management. *Transportation Research Part C: Emerging Technologies*, 77:226–244, 12 2017.

[MBM+16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[MBO15] Daniel Mackowski, Yun Bai, and Yanfeng Ouyang. Parking space man-

agement via dynamic performance-based pricing. *Transportation Research Part C: Emerging Technologies*, 59:66–91, December 2015.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[MMN+19] Elena Mocanu, Decebal Constantin Mocanu, Phuong H. Nguyen, Antonio Liotta, Michael E. Webber, Madeleine Gibescu, and J. G. Slootweg. On-Line Building Energy Optimization Using Deep Reinforcement Learning. *IEEE Transactions on Smart Grid*, 10(4):3698–3708, July 2019.

[NS02] Serguei Netessine and Robert Shumsky. Introduction to the Theory and Practice of Yield Management. *INFORMS Transactions on Education*, 3(1):34–44, September 2002.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RO15] Rupal Rana and Fernando S. Oliveira. Dynamic pricing policies for interdependent perishable products or services using reinforcement learning. *Expert Systems with Applications*, 42(1):426–436, January 2015.

[SASS16] P. Sheelarani, S. P. Anand, S. Shamili, and K. Sruthi. Effective car parking reservation system based on internet of things technologies. In *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pages 1–4, February 2016.

[Sho17] Donald Shoup. *The high cost of free parking: Updated edition*. Routledge, 2017.

[SJLS00] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308, 2000.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[SQAS16] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Poster*, 2016.

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker,

Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Van Den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017.

[STS18] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[SW10] Claude Sammut and Geoffrey I. Webb, editors. *Bellman Equation*, pages 97–97. Springer US, Boston, MA, 2010.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[TYWH18] Qiong Tian, Li Yang, Chenlan Wang, and Hai-Jun Huang. Dynamic pricing for reservation-based parking system: A revenue management method. *Transport Policy*, 71:36–44, 2018.

[vGS16] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press, 2016.

[WSH+16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanc-

tot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1995–2003, 2016.

[XQH10] Xiaoyan Xu, Yuqing Qi, and Zhongsheng Hua. Forecasting demand of commodities after natural disasters. *Expert Systems with Applications*, 37(6):4313–4317, 2010.

[YQC+18] Peng Ye, Julian Qian, Jieying Chen, Chen-hung Wu, Yitong Zhou, Spencer De Mars, Frank Yang, and Li Zhang. Customized regression model for airbnb dynamic pricing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 932–940. ACM, 2018.

[ZG16] Nan Zheng and Nikolas Geroliminis. Modeling and optimization of multimodal urban networks with limited parking and dynamic pricing. *Transportation Research Part B: Methodological*, 83:36–58, December 2016.

[ZKWL15] Bo Zou, Nabin Kafle, Ouri Wolfson, and Jie (Jane) Lin. A mechanism design based approach to solving parking slot assignment in the information era. *Transportation Research Part B: Methodological*, 81:631–653, November 2015.

[ZZ16] Rong Zhang and Lichao Zhu. Curbside parking pricing in a city centre using a threshold. *Transport Policy*, 52:16–27, November 2016.

[ZZZ+18] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A Deep Reinforcement Learning Framework for News Recommendation. *[WWW2018] Proceedings of the 2018 World Wide Web Conference*, 2:167–176, 2018.